CAMBIAMO INSIEME HACKER JOURNAL (IL GIORNALE SIETE VOI)

Vorrei aprire questo numero 204 con un accenno al cambiamento di periodicità di Hacker Journal che, dopo molti anni vissuti da quattordicinale, è diventato un mensile. Questo cambiamento del resto non stupirà più di tanto i lettori perché la decisione è stata presa poco prima di andare in stampa con il numero 203 e ha comportato che i numeri 202 e 203 rimanessero in edicola per un mese esatto. Purtroppo i tempi di lavorazione editoriale mi consentono di fare questa comunicazione solo sul numero 204. Alla base di questa decisione c'è un andamento nelle vendite che negli ultimi mesi, anzi forse sarebbe meglio parlare dell'ultimo anno, non ha rispecchiato le aspettative e la speranza che rimanendo in edicola quattordici giorni in più Hacker Journal possa aumentare le proprie vendite. Non è solo una crisi di HJ, che non è né improvvisa, né inaspettata. E' qualcosa di più. E' la crisi di un mondo, quello editoriale, che sta cercando affannosamente di sopravvivere assediato dalla concorrenza, fortissima, di altri media, Internet su tutti. Non vorrei tediarvi troppo e per darvi uno spaccato della situazione attuale vi rimando volentieri alla nostra rubrica della posta che per un numero si è trasformata, riportando i post più significativi raccolti sul nostro forum relativamente all'argomento in questione.Bene. Chiudiamo qui le brutte notizie e passiamo invece ai contenuti del numero 204 che trovo particolarmente stimolanti. Sta per volgere al termine il nostro corso di programmazione in C che ha raccolto un grosso seguito di lettori appassionati, ma vorrei anche segnalarvi l'articolo che spiega le tecniche per bypassare le chiavi seriali di un videogioco. Credo che molti di noi si siano già cimentati, per scopi puramente didattici, naturalmente, in questo tipo di tecniche, però un bel "ripasso" non guasta… Vi portiamo poi alla scoperta di un ottimo strumento open source e multipiattaforma per gestire in modo ottimale tutte le policies di un firewall, ovvero Firewall Builder. E poi… vi lascio la sorpresa di scoprire da soli gli altri argomenti condensati, come sempre, in 32 pagine ricche di testo e informazioni. Come mi è già capitato di scrivere anche sul forum noi ci siamo stati, ci siamo e, se sarete dalla nostra parte, probabilmente ci saremo…

Altair

laboratorio@hackerjournal.it
Questo indirizzo è stato creato per inviare articoli, codici, spunti e idee. E' quindi proprio una sorta di "incubatore di idee".

posta@hackerjournal.it
E' l'account creato per l'omonima rubrica che è ricomparsa nelle pagine della rivista. A questo indirizzo dovete inviare tutte le mail che volete vengano pubblicate su HJ.

redazione@hackerjournal.it
Questo è l'indirizzo canonico. Quello con cui potete avere un filo diretto, sempre, con la redazione, per qualsiasi motivo che non rientri nelle due precedenti categorie di posta.

Sommario

Anno 10 - N.204
Agosto 2010

Editore (sede legale)
WLF Publishing S.r.l.
Socio Unico Medi & Son S.r.l.
via Donatello 71 - 00196 Roma
Fax 063214606

Realizzazione editoriale
Progetti e promozioni Srl
redazione@progettiepromozioni.com

Printing
Grafiche Mazzucchelli S.p.a - Seriate (BG)

Distributore
M-DIS Distributore SPA
Via Cazzaniga 19 - 20132 Milano

Hacker Journal
Pubblicazione registrata al Tribunale di Milano il 27/10/03 con il numero 601.
Una copia: 2,00 euro

Direttore Responsabile
Teresa Carsaniga
redazione@hackerjournal.it

WLF Publishing S.r.l. - Socio Unico Medi & Son S.r.l. è titolare esclusivo di tutti i diritti di pubblicazione. Per i diritti di riproduzione, l'Editore si dichiara pienamente disponibile a regolare eventuali spettanze per quelle immagini di cui non sia stato possibile reperire la fonte.

Gli articoli contenuti in Hacker Journal hanno scopo prettamente divulgativo. L'Editore declina ogni responsabilità circa l'uso improprio delle tecniche che vengono descritte al suo interno. L'invio di immagini ne autorizza implicitamente la pubblicazione anche non della WLF Publishing S.r.l. - Socio Unico Medi & Son S.r.l.

Copyright WLF Publishing S.r.l.
Tutti i contenuti sono protetti da licenza Creative Commons Attribuzione-Non commerciale-Non opere derivate 2.5 Italia: creativecommons.org/licenses/by-nc-nd/2.5/it

Informativa e Consenso in materia di trattamento dei dati personali (Codice Privacy d.lgs. 196/03)
Nel vigore del d.lgs. 196/03 il Titolare del trattamento dei dati personali, ex art. 28 d.lgs. 196/03 è WLF Publishing S.r.l. - Socio Unico Medi & Son S.r.l. (di seguito anche "Società", e/o "WLF Publishing"), con sede in via Donatello 71 Roma. La stessa La informa che i Suoi dati verranno raccolti, trattati e conservati nel rispetto del decreto legislativo ora enunciato anche per attività connesse all'azienda. La avvisiamo, inoltre, che i Suoi dati potranno essere comunicati e/o trattati nel vigore della Legge, anche all'estero, da società e/o persone che prestano servizi in favore della Società. In ogni momento Lei potrà chiedere la modifica, la correzione e/o la cancellazione dei Suoi dati ovvero esercitare tutti i diritti previsti dagli artt. 7 e ss. del d.lgs. 196/03 mediante comunicazione scritta alla WLF Publishing S.r.l. e/o al personale Incaricato preposto al trattamento dei dati. La lettura della presente informativa deve intendersi quale consenso espresso al trattamento dei dati personali.

Copertina: Daniela Festa ldfesta@libero.it

CAMBIAMENTI AUSPICABILI, NECESSARI,

SPERIAMO UTILI

Questo numero, il 204, di HJ risente un po' della frenesia del momento e dei tanti importanti cambiamenti che dal prossimo numero probabilmente già troverete sulla rivista e che in qualche modo vorremmo anticiparvi proprio in questo spazio. Forse troverete questa pagina un po' dissonante dai toni dell'editoriale, però il motivo è che l'abbiamo allegata proprio all'ultimo momento, appena prima di andare in stampa, quindi è successiva all'editoriale stesso, di cui rappresenta, se volete, un'appendice imprevista ma importante. Importante perché, dopo aver preso coscienza dei problemi, abbiamo cercato di elaborare, insieme all'editore, una linea operativa che ha lo scopo di rilanciare la rivista e che vi rissumiamo nei seguenti punti.

GRAFICA

Il prossimo numero sarà più sobrio. I tanti teschi (che peraltro abbiamo trovato tutti divertenti, specie all'inizio) sembrano non piacere più, così come la grafica troppo vistosa. Molti utenti sul sito denunciano il fatto che i contenuti siano assai più importanti e autorevoli di quanto non suggerisca la grafica. Alcuni responsabili di aziende IT ci segnalano che apprezzano la rivista per i contenuti ma mai si sognerebbero di portarla in ufficio per via proprio della veste grafica.

Si cercherà di darle una veste più "seria", pescando sempre nell'iconografia del mondo digitale, ma senza eccessi.

Partecipazione diretta dei lettori
Tornando in parte alle origini, la rivista si apre ancora di più a contributi esterni, sia di singoli lettori che di realtà più composite. L'intento è quello di rinsaldare i legami con la variegata comunità di sviluppatori, esperti di sicurezza, hacker e frequentatori degli ambienti undergorund e più alternativi della rete. Quindi ci aspettiamo una vostra decisa partecipazione alla vita del giornale. Le modalità ce le potete suggerire anche voi.

FORMATO ELETTRONICO

Abbiamo deciso di creare una versione in pdf, scaricabile a pagamento direttamente dal sito www.hackerjournal.it. Si tratta di un accorgimento che vuole venire incontro a tutti coloro, e sono davvero tanti, che chiedono di potersi abbonare alla rivista, possibilità fino ad ora negata per il formato tradizionale cartaceo, e anche per chi ha difficoltà a trovare Hacker Journal in edicola (sono diversi i lettori che ci segnalano questo problema). Sul sito e nei prossimi numeri della rivista vi avviseremo tempestivamente appena il servizio sarà operativo.

EQUILIBRIO

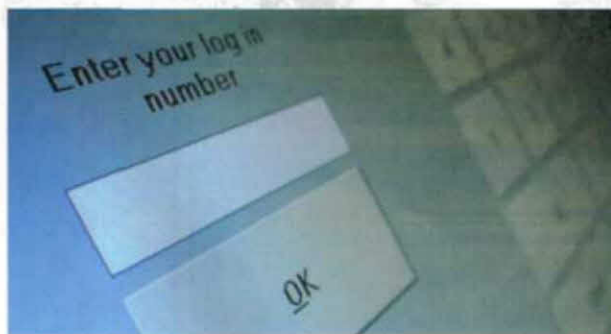
Hacker Journal ha sempre avuto una doppia anima in precario equilibrio tra tecnicismo, con righe di codice poco comprensibili per i neofiti e articoli decisamente più "abbordabili". Negli ultimi tempi abbiamo accentuato questo tema tecnico incontrando il favore di diversi lettori ma, probabilmente, scoraggiandone altri. Cercheremo di spostare un po' gli equilibri in

una linea mediana per provare ad accontentare tutti i nostri lettori. Già in questo numero, tuttavia, troverete un buon mix di articoli più discorsivi contrapposti ad altri decisamente più tecnici.

PIÙ IMPORTANZA ALLA SICUREZZA

Hacker Journal è una rivista dedicata in qualche modo alla sicurezza. Lo è sempre stata. Dove c'è una tecnica di attacco c'è una risposta difensiva. Questo è chiaro. Proprio per questo motivo tra i nostri lettori ci sono molti professionisti che trovano nei nostri articoli degli spunti interessanti. Vogliamo in qualche modo rivolgerci in modo più mirato anche a questo target creando una sezione fissa dedicata proprio alla sicurezza e alle tecniche di difesa. Si tratta di un esperimento che, in base al gradimento dei lettori, potrebbe trasformarsi in un'iniziativa editoriale a se stante. Una sorta di spin off che potrebbe trovare una sua indipendenza da Hacker Journal.

UTENTI POCO CONSAPEVOLI
PASSWORD A RISCHIO

Se le password finiscono nelle mani sbagliate le conseguenze personali ed economiche possono essere disastrose. Nonostante molte persone utilizzino le password in modo responsabile, sono ancora troppi a usarne una sola per qualsiasi tipo di servizio. Secondo una ricerca commissionata da F-Secure, circa il 20% degli utenti Internet in Germania, Regno Unito e Svezia usa un'unica password per tutti i servizi, dall'online banking, alle e-mail, ai videogiochi online. Circa il 20% degli intervistati scrive le proprie password su pezzi di carta, mentre l'8% le dimentica facilmente ed è costretto a cambiarle spesso.
Un'altra indagine di F-Secure condotta in sette Paesi rivela che, in media, solo il 50% di chi utilizza dispositivi mobili protegge il proprio telefono con una password. Secondo lo studio, i tedeschi sono i più consapevoli dal punto di vista della sicurezza, con il 68% che utilizza password, mentre britannici (27%) e americani (13%) restano parecchio indietro in termini di utilizzo sicuro dei dispositivi mobili.
"Trovandosi oggi a gestire un numero considerevole di dati per l'accesso online, gli utenti spesso cadono nella tentazione di usare solo una o due password per qualsiasi servizio", ha dichiarato Sean Sullivan, Security Advisor di F-Secure."Sfortunatamente, si tratta di un metodo che può portare a spiacevoli conseguenze, poiché molti sono cybercriminali che cercano continuamente nuovi modi per rubare le password e sfruttarle a loro vantaggio". Da parecchio tempo i criminali informatici si servono di false email per chiedere la conferma di password e nomi utente e accedere quindi ai dati bancari e ad altre informazioni personali.
Con una crescita così rapida, anche Facebook è diventato uno degli obiettivi preferiti da cybercriminali per sottrarre password agli utenti. Compromettendo gli account di Facebook, i criminali puntano ad accedere alle email degli utenti, raggiungendo così una notevole quantità di dati riservati. Le persone che utilizzano la stessa password per tutto hanno quindi molto di più da perdere.
"Per le password, non usare mai informazioni presenti nei profili su Facebook, come la data di nascita, il proprio nome o quello dell'animale domestico", ha aggiunto Sean Sullivan. "Raccomando a tutti di dedicare pochi minuti per apprendere un sistema in grado di creare password uniche, particolarmente critiche per tutti quei servizi online che rendono disponibili informazioni sugli utenti".
Mischiare lettere e numeri è un buon modo per creare password più sicure. Ma è possibile ricordare tante password quanti sono i siti web che visitiamo? Ecco un metodo semplice illustrato sul blog Safe and Savvy di F-Secure:http://safeandsavvy.f-secure.com/2010/03/15/how-to-create-and-remember-strong-passwords/
Utilizzare password sicure, cancellare e-mail riservate ed essere consapevoli di come si muovono i criminali online sono ottime abitudini che tutti possiamo mettere in pratica. Inoltre, è buona norma utilizzare un unico account email per le transazioni "business", ad esempio per tutto ciò che riguarda la banca, e un altro per registrarsi ai vari servizi online come Facebook o il portale di news preferito. E, naturalmente, utilizzare software di sicurezza in grado di bloccare gli attacchi dei virus che provano a infettare i computer.

ATTACCO HACKER ALL'IPAD: AT&T CI METTE UNA PEZZA!

AT & T Inc ha dichiarato mercoledì, in un comunicato stampa, che alcuni utenti di Apple iPad hanno avuto informazioni personali esposte tramite una falla di sicurezza di rete.

La violazione, segnalata in primo luogo dal sito Gawker, ha portato alla esposizione solo di indirizzi email.

AT & T si è scusata e ha detto che informerà tutti i clienti che sono stati colpiti dalla violazione della sicurezza che si è verificata lunedì ed è stato prontamente corretta.

Apple, dal canto suo, non ha rilasciato alcuna dichiarazione.

Nella relazione di AT & T risulta che più di 100.000 account utente iPad possono essere stati compromessi. Secondo il sito Gawker tra questi figurerebbero indirizzi e-mail che comprendono celebrità, politici e dirigenti.

AT & T è attualmente l'unico carrier wireless per i iPad e iPhone negli Stati Uniti ma continua ad attirare le critiche aspre da parte degli utenti iPhone sulla qualità della sua rete, e la violazione di sicurezza riconosciuta mercoledì non può che aggiungere benzina sul fuoco.

Il vettore ha detto che il difetto ha portato all'esposizione di IDs del circuito integrato che identifica la SIM nei dispositivi mobili.

Secondo AT & T: "L'unica informazione che può essere derivata dalla IDs è l'indirizzo email collegato a tale dispositivo".

Gli analisti ipotizzano che i "rivali" di Verizon Wireless potrebbe eventualmente spodestare AT & T come vettore principale per l'iPhone di Apple forse già nel 2011.

E' EXPLOITS
IL MALWARE PIÙ USATO DAI CRIMINALI-INFORMATICI

Kaspersky Lab ha emesso il suo report sulle statistiche dei malware specificando quali infezioni sono state riscontrate e bloccate dalle soluzioni Kaspersky nel mese di maggio. Exploits - un programma dedicato per l'attacco attraverso alcune vulnerabilità di software regolari - ed i suoi relativi trojan, hanno dominato non solo la top 20 dei malware rilevati sui computer degli utenti, ma anche la classifica dei malware nativi su Internet.

Negli ultimi mesi, entrambe le classifiche hanno mostrato un deciso incremento dell'uso di Exploit da parte dei criminali informatici. Il loro obiettivo rimane la stesso - il furto di dati sensibili dell'utente - ma le tecniche di propagazione ed i metodi per prevenire l'identificazione ed il rilevamento sono cambiati.

Una voce degna di nota nella top 20 dei malware è un trojan che ruba dati di accesso, account e password, dei più popolari giochi online. I giocatori di CabalOnline, Metin2, Mu online e vari altri giochi sviluppati da Nexon. net sono stati tutti colpiti dal "Trojan-GameThief.Win32. Magania.dbtv".

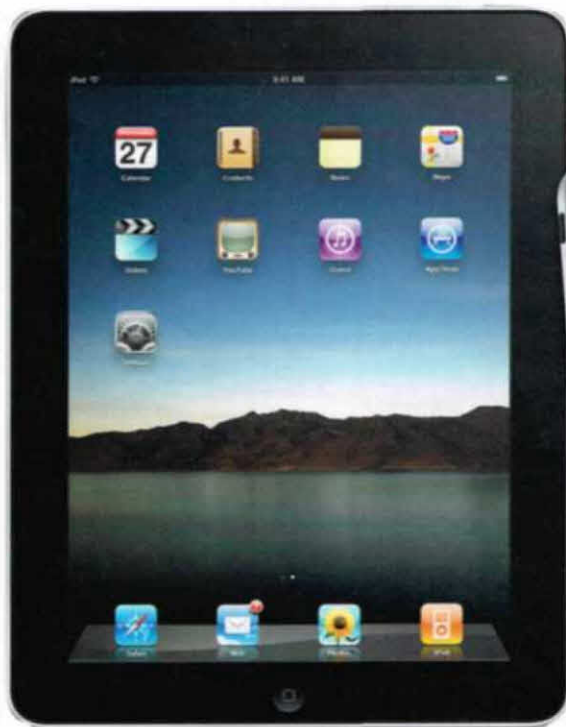
Undici dei malware della top 20 di maggio sono Exploit, di un tipo o dell'altro, ed i suoi relativi trojan. Questi programmi maligni occupano cinque posti consecutivi nella top 20 a partire dal 2° posto, comparendo nella lista in gruppi di due o tre.

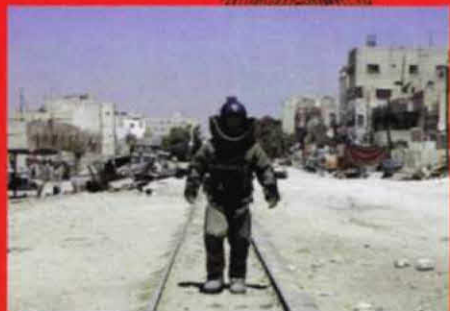
Tre dei nuovi arrivati sono Exploit per Java e pertanto si consiglia vivamente agli utenti di questa piattaforma di verificare spesso la presenza di aggiornamenti del software.

Il prima posto nella top 20 va al "Trojan-Clicker.JS.Iframe. bb". Questo particolare malware è progettato per aumentare il contatore delle visite ad un determinato sito, inducendo il computer infettato a generare visite all'insaputa e senza il consenso del proprietario. Nel mese di maggio solo questo trojan ha infettato quasi 400.000 siti web.

La versione completa delle statistiche di maggio di Kaspersky Lab sono disponibili all'indirizzo www. securelist.com o sul sito italiano http://www.kaspersky. com/it/news?id=248

Un maleware per iPhone insidia anche l'iPad

Lo scorso anno, i laboratori di Panda avevano segnalato la presenza del worm iPhone/Eeeki, capace di colpire i dispositivi modificati dagli utenti (per riuscire a installare applicazioni non ufficiali). Questo codice era in grado di diffondersi anche su iPod Touch.
Nonostante Apple abbia deciso di bloccare completamente l'hardware - rendendo impossibile installare periferiche - e il software, in quanto tutte le applicazioni si installano dall'App Store della casa produttrice, i cyber criminali hanno trovato un metodo per colpire i dispositivi manomessi. Il malware progettato per iPhone ha la stessa capacità di attaccare e diffondersi sui dispositivi iPad, in quanto entrambi i prodotti possiedono il medesimo sistema operativo, noto come iPhone (v3) o iOS (v4) della prossima versione.
Luis Corrons, direttore tecnico dei laboratori di Panda Security, spiega: "Tutto questo non significa che assisteremo a un'ondata di attacchi. Da sempre siamo consapevoli che la crescita nell'utilizzo dei dispositivi Apple avrebbe portato i cyber criminali a sviluppare un metodo per colpire gli utenti di queste piattaforme. Stiamo assistendo, senza dubbio, a numerosi tentativi "teorici", quindi raccomandiamo agli utenti Apple di seguire le indicazioni del produttore per garantire la massima sicurezza ai propri sistemi operativi".

HURT LOCKER DÀ BATTAGLIA AL FILE SHARING

La società di produzione del film premio Oscar Hurt Locker ha intrapreso una vera e propria battaglia legale contro coloro che scaricano film illegalmente. L'Internet Service Provider ha individuato 5.000 connessioni di rete coinvolti nella condivisione del film attraverso le reti peer-to-peer, successivamente dietro richiesta della società tali connessioni sarebbero state identificate per poter procedere con la causa legale.
In questo modo la Voltage Pictures vuole dare un segnale forte per contrastare il fenomeno dilagante del file-sharing che, secondo la società di produzione, oltre a contribuire al declino del settore cinematografico, danneggia economicamente tutti coloro che in quel settore operano.
Di parere contrario è invece Peter Sunde, uno dei membri fondatori del tanto discusso sito di file sharing "Pirate Bay" che ha dichiarato che il file sharing aumenta le entrate e l'interesse dei film o dei brani musicali scaricati e che trova stupido intentare cause legali contro i propri fan.

BOT PROGETTATI SU COMMISSIONE

E' stata rilevata e sgominata una rete di vendita di bot creati per colpire i social network e i sistemi webmail, tra i quali Twitter, Facebook, Hi5, MySpace, MyYearBook, YouTube, Tuenti, Friendster, Gmail e Yahoo. La pagina, pubblicata su Internet, mette a disposizione un ricco catalogo di programmi e ogni voce spiega la motivazione per la quale il bot è stato sviluppato: la creazione di account multipli simultanei sui social network, il furto o l'appropriazione di identità di amici e conoscenti, l'invio automatico di messaggi, etc... Secondo quanto affermato sul sito "Tutti i bot operano in modo convenzionale: raccolgono ID o nomi di amici e inviano richieste di amicizia, messaggi e commenti in maniera automatica".
Per quanto riguarda il listino, il bot più economico ha un costo di 95 dollari, mentre il più costoso arriva a 225 dollari. Il catalogo completo può essere acquistato a 4.500 dollari, con la garanzia che i bot non potranno essere rilevati da nessun software, grazie alla modifica, quando necessario, di utenti, agenti e intestazioni per evitare che vengano bloccati. Inoltre, eludono il meccanismo di sicurezza CAPTCHA, incluso su molti siti, così l'acquirente dovrà solo selezionare i parametri e lasciare che i bot operino da soli, con la garanzia di aggiornamenti continui.

Ecco alcune delle inconsuete attività per le quali sono stati progettati i bot:
- Creazione automatica di visite e visualizzazioni di video su YouTube
- Gestione dei ranking su Alexa - Manipolazione delle preferenze sul sito web Digg
- Invio illimitato di messaggi sui siti che organizzano gli incontri online, come DirectMatches.

Social Networking Bots

Winsock Bots (For MySpace, Hi5, Tagged and much more)

MySpace Bots

Special Offer: Buy all of the winsock bots for $4500 only

Accounts Creator

NoeXKuzE
redazione@hackerjournal.it

Crack di iOS 4

Qualcuno già li conoscerà, per gli altri diventeranno presto familiari, dato che i ragazzi del Dev Team dopo aver sbloccato in passato gli iPhone dalla prima alla terza generazione, si sono occupati rapidamente di rimuovere il blocco anche da **iOS 4**, il sistema operativo dell'iPhone 4. Pochissimo tempo dopo il rilascio del sistema operativo, hanno infatti reso disponibile nel loro blog (http://blog.iphone-dev.org) l'aggiornamento di **PwnageTool**, uno strumento in grado di modificare il firmware originale per permettere di aggiornare iPhone 3G e 3GS già sbloccati alla nuova versione di iOS 4 senza perdere precedenti sblocchi. Lo sblocco in questione riguarda la possibilità di installare applicazioni liberamente e viene chiamato candidamente "Jailbroken", ossia "gabbia rotta". Vediamo cosa si deve fare!

CRACKING

COME AVERE IOS 4 SU IPHONE 3GS SENZA I BLOCCHI DI APPLE.

L'iPhone 4 è lo smartphone più venduto al mondo. Peccato che iOS 4 sia un po' troppo chiuso per chi come noi ama aprire i giocattoli e farli funzionare diversamente!

REQUISITI

Come premessa dovuta, ricordiamoci che modificare il firmware del telefonino in modo non ufficiale può invalidare la garanzia ed è bene che tale operazione venga comunque fatta solo se ci si rende bene conto di cosa stiamo facendo.
Detto ciò, al momento PwnageTool 4.01 è disponibile solo per Mac (con OS X 10.4 +), mentre per Windows (XP/Vista/Seven) dovremo aspettare l'ulteriore sviluppo di **sn0wbreeze** (http://ih8sn0w.com) che permette di avere il firmware Jailbroken, ma non supporta ancora iOS 4.
Va detto subito che iPhone 2G e gli iPod Touch di prima generazione non sono al momento supportati dai tool. Sono invece supportati:
gli iPhone 3GS che abbiano già a bordo un firmware "Jailbroken" e con bootrom più vecchio (per capirci, se avete iBoot-359.3.2 o successivi non avete per ora alcuna possibilità perché con quell'aggiornamento Apple ha chiuso la falla che permetteva di generare uno stack overflow al boot del terminale; stessa storia se avete un iPod Touch di seconda generazione con numero di serie che comincia con "MC" o un iPod Touch 3G;
gli iPhone 3G e iPod Touch con firmware 3.0 e 3.1.2 sui quali si può agire tramite un altro tool del Dev Team chiamato redsn0w 0.9.5 BETA (http://wikee.iphwn.org/howto:rs9) che gira sia su Windows che Mac.
Per sapere che versione di iBoot abbiamo possiamo usare l'utile tool iDetector (http://ih8sn0w.com/index.php/products/view/idetector.snow).
Ovviamente poi il tool non può

funzionare se il terminale è stato già aggiornato in modo standard a iOS 4!

ROMPIAMO LA GABBIA

Supponendo di avere un iPhone 3GS sul quale sperimentare il tool del Dev Team, scarichiamo l'ultima release di iOS4 adatta al nostro telefonino. Abbiamo due possibilità: googliamo "iOS 4 download links" e scegliamo tra i numerosi risultati oppure colleghiamo il 3GS al Mac con il cavo usb e richiediamo il download via iTunes, senza installare o aggiornare nulla (fondamentale!)
Poi prima di proseguire facciamo un bel backup con iTunes seguendo le indicazioni di Apple (http://support.apple.com/kb/HT1766).
PwnageTool è stato creato per modificare il firmware ufficiale (iPhone2,1_4.0_8A293_Restore.ipsw) creandone uno modificato (patchato) proprio nella parte di boot: questa modifica permette di evitare che il terminale compia dei controlli sull'autenticità del firmware e possa essere possibile così evitare i blocchi presenti nel codice dato che aggiungeremo altri pacchetti software, non autorizzati (chissà perché!) da Apple.
Installiamo PwnageTool e verrà creata un'icona che serve per lanciare il programma. Clicchiamoci sopra e aspettiamo che il programma si carichi e visualizzi il menu con 4 icone: selezioniamo l'icona di Einstein per impostare la modalità avanzata. Ci verrà quindi richiesto di indicare quale terminale vogliamo connettere tra iPhone 3GS e iPad Touch 2G. Selezioniamo 3GS e ci verrà chiesto ora di selezionare il firmware: indichiamo il path del file ipsw.
Siamo pronti ora per "personalizzare" il nostro firmware: clicchiamo su "General" e possiamo scegliere di impostare "Activate the phone" se utilizziamo un gestore telefonico diverso da quello che ci ha venduto l'iPhone (altrimenti lasciamo l'icona de-selezionata).
In "Cydia Settings"clicchiamo su "Download packages" e premiamo il pulsante Refresh. Selezioniamo poi I pacchetti che ci interessano (es. OpenSSH e OpenSSL) e accettiamo cliccando sulla freccia blu. Clicchiamo ora su "Select Packages" poi "Select All" e di nuovo sulla freccia blu.
In "Custom Packages Settings" clicchiamo direttamente sulla freccia blu. In "Custom logos settings" abbiamo la possibilità, se ci interessa, di cambiare I loghi visualizzati durante il boot e durante il ripristino del terminale. Per chi vuole disegnarsele, le dimensioni sono 320x480 e va impostata la scala di grigi in RGB.
Ora possiamo cliccare su "Build" e attendere la creazione del firmware modificato. Ci verrà chiesto il nome da dare al file e ci vorranno poi circa 10-30 minuti perché si completi il processo a seconda della configurazione scelta.
Quando il firmware patchato è stato creato, il software ci chiede l'autorizzazione per mettere l'iPhone in modalità ripristino (Recovery mode): diamo ok e connettiamo l'iPhone da spento. A questo punto premiamo e teniamo premuti il tasto Home e quello di Sleep/Wake finché lo schermo diventerà tutto bianco (dopo circa 5 secondi). Lasciamo quindi solo il pulsante di Sleep/Wake, mantenendo premuto il tasto Home finché lo schermo diventerà tutto nero.
A questo punto iTunes ci avviserà con un messaggio che ha rilevato un iPhone in modalità di ripristino. Diamo ok e il nostro iPhone è pronto per l'aggiornamento.
A questo punto dobbiamo procedere con molta attenzione: tenendo premuto il tasto Alt/Options premiamo sul pulsante "Restore". Questo è il punto più delicato di tutto il processo: se non teniamo premuto Alt/Options ci ritroveremo l'iPhone aggiornato a iOS 4, ma completamente bloccato e senza possibilità di tornare indietro. Invece tenendolo premuto ci verrà chiesto il percorso del firmware che vogliamo flashare. Selezioniamo il nostro firmware personalizzato e attendiamo che iPhone si occupi del trasferimento che durerà altri 10 minuti.
Una volta flashato, avremo un 3GS con iOS4 Jailbroken, pronto per tutte le nostre sperimentazioni! E non dimentichiamo di ripristinare il backup fatto prima, così da tornare immediatamente operativi.

Scarichiamo dal sito del Dev Team PwnageTool direttamente in una cartella "Pwnage" sul desktop.

Per ora il software supporta solamente iPhone 3gS e iPad Touch 2G.

Dopo aver selezionato tutti I pacchetti che ci interessano possiamo procedere con la creazione del firmware modificato

Il processo di creazione del nuovo firmware può essere lungo, arrivando anche a 45 minuti.

di penelope.di.pixel
redazione@hackerjournal.it

R4: ISTRUZIONI PER L'USO

HACKING

INSTALLARE FILE .NDS SUL NINTENDO DS O DS LITE? NULLA DI PIÙ SEMPLICE CON LA SCHEDA R4.

Molti lettori ci hanno chiesto delucidazioni sulla scheda R4 Revolution per Nintendo DS Lite. Si tratta di un dispositivo in commercio che consente la lettura dei file. nds, quindi dà, in parole povere, la possibilità di caricare e fare girare giochi per Nintendo DS. Per il nostro test abbiamo acquistato in un negozio di Milano una scheda R4 SDHC, upgrade della Revolution, con micro sd da 2GB: costo 32 euro. La confezione contiene:
La scheda R4 da inserire nell'alloggiamento del Nintendo DS riservato alle cartucce di gioco.
Un adattatore USB in cui inserire la micro DS per visualizzarla sul PC e caricarvi i contenuti.
Una scheda micro Sd da 2 GB (ma il taglio può essere anche superiore).

L'uso della R4 SDHC è piuttosto semplice. Basta caricare un gioco con estensione .nds sulla scheda micro SD, inserirla nell'alloggiamento della R4 e, infine, quest'ultima nel Nintendo DS e avviare.
Prima di fare ciò bisogna però assicurarsi di caricare il kernel corretto, ovvero il sistema operativo in grado di installare l'interfaccia e fare girare tutti i componenti.

Per fare ciò basta collegarsi solitamente al sito del produttore, nel nostro caso http://www.r4i-sdhc.com/indexe.asp, scaricare il kernel appropriato (nella sezione download) alla scheda R4 tra i diversi disponibili, quindi scompattarlo e caricare tutti i file contenuti nella directory principale della scheda SD. Nel nostro caso abbiamo caricato i seguenti file:

Moonmemo (cartella)
Moonshl (cartella)
R4iMenu (cartella)
R4.dat

Ora la R4 è pronta per ospitare i giochi Nintendo con estensione .nds. Avviando il Nintendo DS dopo averla inserita verrà visualizzata un'interfaccia grafica che mostra l'elenco di tutte le risorse .nds, ovvero dei giochi, disponibili. Basta selezionare un gioco ed avviare, la scheda genererà il file di salvataggio relativo e si potrà iniziare giocare.

PROBLEMI DIFFUSI

Uno dei problemi più diffusi è il mancato caricamento dell'interfaccia o del gioco. Ovvero il sistema si blocca in un "loading" che non finisce mai. Si tratta in questo caso di un problema di kernel, probabilmente avete scaricato e installato un kernel non adatto a quella scheda R4. Per risolvere il problema basta scaricare la versione corretta e tutto andrà a posto.

Un altro problema decisamente più fastidioso è il sistema operativo che va in "crash" dopo alcuni giorni che non si utilizza la cartuccia R4 impedendo il caricamento dell'interfaccia e l'utilizzo dei giochi. Si tratta di un problema noto che può essere risolto semplicemente spostando indietro di qualche giorno la data del proprio Nintendo DS. E' un'operazione un po' empirica ma funziona. Evidentemente questo accorgimento è piuttosto sgradito a chi gioca con titoli come Animal Crossing che fanno progredire il gioco in base al calendario.

IL FORUM

IN EVIDENZA

Hacker Journal dal numero 203 è diventato mensile. Questa la notizia che molti di voi già sanno o hanno intuito e che vi confermiamo proprio in questa sede. Al di là dei molti perché ci sembrava giusto riportare alcune delle riflessioni più importanti che sono scaturite nel forum del nostro sito www.hackerjournal.it. Già perché sul sito abbiamo dato la notizia con una certa tempestività è c'è stato tutto il tempo per reagire in modo emotivo, riflettere e fare sedimentare le cose. Ne sono scaturite idee, indicazioni, commenti, critiche e elogi che vogliamo in parte trasmettere anche a tutti i lettori della rivista che non sono abituali frequentatori del sito.
Riteniamo che uno dei grandi elementi di forza di Hacker Journal sia proprio la sua comunità di lettori e utenti del sito. Ci piacerebbe quindi che queste due realtà si unissero in un unico fronte animato dalla voglia di spingere Hacker Journal verso risultati sempre migliori.
Gli utenti del sito si sono pronunciati e si stanno tutt'ora pronunciando. Ora, cari lettori, è il vostro turno. Le vostre indicazioni alimenteranno la rubrica della posta che tornerà al suo posto già dal prossimo numero dopo avere ceduto lo spazio, per un mese, alle voci "della rete".

[IMPORTANTE] HACKER JOURNAL DIVENTA MENSILE.
da altair » 18 giu 2010, 18:38

Forse qualcuno di voi passando davanti all'edicola si sarà chiesto che fine avesse fatto il numero 203 di HJ. Domanda legittima, infatti il numero 203 sta per uscire... dopo un mese dall'uscita del 202. Già, l'avrete sicuramente capito, Hacker Journal ha cambiato periodicità, è diventato un mensile. Inutile che vi racconti che è una precisa strategia di marketing o qualche altra fesseria girata in chiave positiva, la realtà è che la rivista risente della grande crisi editoriale, non da ora, ma da un po' di tempo a questa parte. Il nostro sforzo, specie nell'ultimo periodo, è stato quello di cercare di farla crescere (come lettori) ma purtroppo i dati di vendita sono rimasti stabili. Da qui la decisione dell'editore.
In questa notizia non certo positiva si possono tuttavia trovare degli spunti di ottimismo (guai a non cercarli). Il sito va molto bene, è in crescita esponenziale, la rivista pur non aumentando le vendite non arretra e, ultima notizia, presto sarà disponibile in formato elettronico per iPad (occorrerà scaricare la relativa applicazione).
La mia idea è che HJ abbia un futuro. Noi ci crediamo. Quindi continueremo ad esserci finché l'editore lo vorrà. La mia speranza è che la crescita del sito possa in qualche modo trascinare al rialzo le vendite della rivista, ma è solo una speranza. Comunque una grossa mano, come sempre, potete senz'altro darcela voi divulgando la nostra passione per un aspetto dell'informatica forse di nicchia, ma che merita, a mio avviso, di continuare a fare sentire la sua voce.

Re: Hacker Journal diventa mensile
da BlackLight » 18 giu 2010, 20:59

altair ha scritto:
presto sarà disponibile in formato elettronico per iPad (occorrerà scaricare la relativa applicazione).

Quando mi avevano accennato a questa notizia speravo che non fosse vera...sinceramente

1. quale percentuale dei lettori o potenziali lettori di HJ comprerà un iPad
2. quale percentuale di questi lettori scaricherà un'applicazione per iPad apposta per leggere HJ sul loro gioiellino inutile, magari pagando
3. quale percentuale dei lettori illustrati sopra andrà in giro con il proprio iPad leggendo HJ in quel formato in metro, aspettando il bus o in spiaggia

Fra le varie scelte editoriali possibili, questa è senz'altro la peggiore, che comporterà un investimento che ripagherà in modo misero la casa editrice.
Ci potevano essere scelte migliori per rilanciare la rivista. Fra queste quella di farla diventare una eZine "a metà", magari pubblicando sul web ogni mese un articolo tratto dal numero corrente, o l'introduzione o parte degli articoli sulla rivista, in modo da stuzzicare l'appetito dei lettori e fargli venire voglia di andare in edicola per leggere il resto del numero. Questa è una scelta a mio parere costruttiva, non quella di rendere la rivista disponibile solo per un aggeggio di cui non si conoscono ancora i dati di diffusione e soprattutto non si sa se sarà usato principalmente per la lettura. Questo è quello che io chiamo un salto nel vuoto che può comportare un suicidio strategico inutile.
Altra cosa, non si può parlare della sicurezza di OpenBSD o di programmazione su una rivista che sembra Becchini Domani. Non so se il grafico con tutto il rispetto ha un passato da necrofilo, ma parlare di temi informatici di livello anche alto su una rivista piena di teschi, diavoletti e boiate varie, fa ridere. Gli argomenti trattati potrebbero interessare anche una fascia di lettori più "professionale" rispetto al ragazzino, ma ce lo vedete il professionista di IT security che va in azienda con una rivista piena di teschietti?

Re: Hacker Journal diventa mensile
da ray » 19 giu 2010, 03:01

Che la rivista diventi mensile, non mi dispiace troppo. Magari restando una settimana in più in edicola si limitano i "resi", e magari si potrebbero aumentare le pagine. Per quanto riguarda la versione elettronica,

IL FORUM

anche io penso sia una pessima idea.. aggiungo la motivazione "pirateria". Chi comprerebbe più hj se il primo che la scarica la rende disponibile sul suo blog? Stessa cosa per quanto riguarda la grafica davvero imbarazzante.. la penso come black. Ma dove li trova tutti quei teschi il grafico?

Re: [IMPORTANTE] Hacker Journal diventa mensile.
da jackmasson » 19 giu 2010, 21:23

Da qualche tempo la rivista ha un acquirente in più... me, anche se non credo che basti ad impennare i picchi di vendita...
Però penso che la rivista sia ottima, certo, per me che lavoro (fortunatamente solo lavoro) a Palermo, se non è un'edicola è l'altra, la trovo.
Sono molto favorevole alla versione .pdf, da ricevere sulla casella mail pagando l'abbonamento, visto che non è possibile averlo in formato cartaceo, certo, il rischio che qualcuno lo pubblichi sul proprio blog è concreto, ma sarebbe un pò fessacchiotto, ma come, io pago per avere qualcosa e poi lo distribuisco gratis a tutti? Purtroppo c'è anche di questa gente...
La veste grafica secondo me non riveste tutta questa importanza: voglio dire, chi conosce la rivista (anche il professionista di IT) sa cosa c'è all'interno, anche se la copertina sembra fatta per attirare i ragazzini, si è vero, l'occhio vuole la sua parte, ma che non vada a scapito della sostanza...

Re: [IMPORTANTE] Hacker Journal diventa mensile.
da midKnight » 21 giu 2010, 14:27

Leggo solo ora, anche perché è tempo che non leggo/scrivo qui sul forum.

- L'idea dell'uscita mensile non è neanche malaccio, in effetti: in un'altra occasione almeno ne avevo parlato, senza la pretesa di "suggerire" qualcosa a chicchessia;

- Il formato elettronico, onestamente, non ce lo vedo proprio, quantomeno perché mi sembra (come anche Blacklight ha detto) molto dissonante con l'idea di "risparmio" che si vorrebbe dare a questo cambio di rotta, per quale vantaggio poi? Molto fumoso, debbo dire...

- La grafica è effettivamente abbastanza infantile: sebbene sia chiaramente più importante il contenuto, qualcuno ha fatto giustamente notare che un professionista IT, giacca e cravatta magari nell'azienda per cui lavora, non fa un figurone con la rivista piena di teschi in mano...si potrebbe magari rendere le cose un po' più ordinate e "professionali".

Detto questo, credo proprio che sottoscriverei volentieri l'abbonamento ad un pdf mensile.

Re: [IMPORTANTE] Hacker Journal diventa mensile.
da grv » 28 giu 2010, 19:29

Dico la mia:
la grafica della rivista è totalmente inappropriata.Voi ne fate una questione di target, posso capirlo.
Fosse per me, la rivista sarebbe a font più piccoli mantenendo leggibilità, grglie ottimizzate ad utilizzare tutto lo spazio possibile. Strizzerei l'occhio al carattere nerd degli utenti, spingerei sulla ricerca grafica in pieno stile DEMOSCENA, unita però alla grinta di contenuti d'avanguardia.

ZERO teschi e molti, ma MOLTI più contenuti. Non necessariamente degli articoli completi che, di fatto, ci sono.

Ma molte più news, più novità dal mondo dell'informatica, del nuovo hardware, delle nuove tecnologie e del software. Bastano poche righe, ed il necessario link di riferimento al web, per scatenare la curiosità del nerd che compra la rivista soprattutto per avere le informazioni che non riesce autonomamente a trovare.
Essere hacker significa essere curiosi. E la curiosità abbraccia tutta la conoscenza umana, non solo l'informatica. Mi piacerebbe allora leggere articoli strani che mi fanno vibrare il cervello. L'arte del modding, del retrocomputing, ma anche di come costruire qualcosa di imprevisto con ciò che si trova nel mercato. Il modellismo, la net art, la matematica che si mischia con la politica e la filosofia e la denuncia. Gli argomenti sarebbero tanti, voi ci avete provato alcune volte ci siete riusciti altre volte meno.

Il massimo della denuncia (sotto i baffi) che ho letto è come Microsoft o Apple siano "stronze". Cerchiamo di capire perchè lo sono, che cosa invece hanno fatto di buono. E gli argomenti non finirebbero qui: Obama vuole un bottone per spegnere internet a livello nazionale.

SPIEGATEMI PERCHÉ. Illuminatemi.

Grazie per ciò che avete fatto sinora, ma per quel che mi riguarda ho una richiesta retorica:
voglio sapere di più di Guglielmo Marconi, come la sua ricerca ha influenzato l'oggi.
Non tentate di targetizzarmi. Incuriositemi ed aiutate la mia curiosità.
In tal caso sono disposto a comprare anche se la rivista ne costasse 10 di euro, perchè ne vale la pena.

di Massimiliano Brasile
redazione@hackerjournal.it

BYPASSARE LA RICHIESTA DI SERIAL DI UN GIOCO

HACKING

LE TECNICHE DI PROTEZIONE DEI GIOCHI SI STANNO EVOLVENDO, MA LA CHIAVE SERIALE È SEMPRE DI GRAN MODA, VEDIAMO COME AGGIRARLA.

SCORE<1> HI-SCORE SCORE<2>
0000 0000
3 CREDIT 00

Lo scopo di questo articolo, che ha una funzione puramente didattica e si rivolge anche a coloro che programmano protezioni per i giochi che sviluppano, è quello di spiegare come bypassare la richiesta di inserimento di un codice seriale da parte di un gioco. Il gioco in questione si chiama QBob, risale a qualche anno fa (una decina per la precisione) e richiede una registrazione per la somma di 20 $ per poter avere più livelli e altro. Riprendendolo dopo un po' di tempo ho deciso di provare a riversarlo per evitare di pagare la registrazione (a un team di programmatori che molto probabilmente non esiste nemmeno più, almeno sotto il nome di MoonRock Software Inc.).

2. GLI STRUMENTI

Per il nostro scopo avremo bisogno di poco software, che elenchiamo di seguito:

- QBob: ovviamente, per riversarlo, avremo bisogno del gioco stesso, che possiamo trovare in versione demo a http://www.moonrock.com/qbob3214.exe .
-W32Dasm: nell'articolo verrà usato w32dasm come disassembler, ma potete usare il vostro preferito senza problemi. Per scaricare W32Dasm il link è http://download.famouswhy.com/software/w32dsm87.zip
- HxD: come per W32Dasm, come editor esadecimale nell'articolo verrà usato HxD per il semplice fatto che è il primo editor esadecimale che ho trovato per Windows cercando su google, qualunque altro editor vi offrirà le stesse performance. HxD lo trovate sul sito http://hxd.softonic.it/ .

Il disassembler ci servirà per visualizzare il contenuto dell'eseguibile di QBob nelle sue istruzioni in assembly, in modo da renderci la vita un po' più semplice invece di andare a leggere l'eseguibile in binario. Una volta che avremo individuato la porzione di programma da modificare lo andremo a fare utilizzando l'editor esadecimale, che consente la modifica di file binari.

3. ALCUNE BASI PRIMA DI INIZIARE

Per comprendere la parte seguente dell'articolo è bene avere alcune conoscenze del campo in cui andremo a lavorare, ossia l'assembly e il reverse engineering. L'assembly è il linguaggio più vicino al linguaggio macchina. Quello che andremo a fare sarà, utilizzando un disassembler, leggere il listato in assembly, trovare la parte di istruzioni che si occupa del controllo del codice seriale e modificarlo in modo da permetterci di inserire qualunque seriale e farlo accettare. Il lavoro del disassembler è quello di prendere i singoli byte che formano un file binario e convertirli nella stringa a cui è convenzionalmente assegnato il dato byte; per esempio invece di andare a leggere 01010000 leggeremo 'push eax', che è un'istruzione utilizzata per inserire nello stack il contenuto del registro eax. Nell'esempio 01010000, corrispondente a 0x50, è l'opcode relativo all'istruzione 'push

eax', quindi se noi, per esempio, volessimo modificare il programma mettendo nello stack il contenuto di ecx invece che quello di eax dovremmo modificare l'istruzione in 'push ecx', che ha come opcode 0x51: andremo quindi a cercare con l'editor esadecimale il byte 0x50 che ci interessa e cambiarlo con 0x51. È importante tenere a mente questo metodo, dato che andremo a usarlo dopo per modificare il nostro eseguibile. Ora andiamo a vedere un'istruzione particolare, ossia JMP, che, come si può capire dal nome (Jump) effettua un "salto" di un numero di byte determinato. Questo tipo di salto può essere incondizionato o condizionato.Il primo è identificato dall'istruzione jmp, mentre quelli condizionati hanno differenti istruzioni a seconda della condizione per cui avviene il salto. I salti non condizionati saltano in qualunque situazione, se per esempio abbiamo questo listato:

```
mov   S1 %eax   mette il valore 1 nel registro eax
cmp   S1 %eax   confronta il registro eax con il valore 1
jmp   lab2      avviene un salto a lab2 qualunque sia l'esito del confronto
lab1:
mov   S2 %eax   quest'istruzione non verrà eseguita
lab2:
inc   %eax      eax, ha valore 1, viene incrementato, avrà quindi il valore 2 al suo interno
```

(mi scuso se la sintassi è AT&T ma è l'unica che sono in grado di scrivere :) Il codice è già commentato, quindi non c'è molto da dire, vediamo che il jump non condizionato salterà alcune istruzioni. Se invece abbiamo:

```
mov   S1,   %eax
cmp   S1,   %eax
jne   lab2
lab1:
mov   S2,   %eax
lab2:
inc   %eax
```

In questo caso notiamo che, al posto di una jump non condizionata ne abbiamo una condizionata,nello specifico una Jump if Not Equal (JNE).Questo tipo di jump salta solo se il confronto avvenuto in precedenza non è uguale, nel nostro caso, dato che abbiamo messo in eax il valore 1 e poi il registro è confrontato con il valore ,1 il salto non verrà effettuato e si proseguirà quindi con l'istruzione 'mov S2, %eax'. Ciò significa che alla fine del listato il registro eax avrà come valore 3. Altri tipi di salti condizionati sono JE (Jump if Equal), JZ (Jump if Zero), JNZ (Jump if Not Zero),JG (Jump if Grater), JGE (Jump if Greater or Equal), JL (Jump if Less), e molti altri... Per chi conoscesse un linguaggio di alto livello (per esempio il C) avrà capito che con questi svariati Jump è possibile gestire tutti i tipi di cicli/condizioni che si utilizzano negli altri linguaggi, per esempio un ciclo for non è altro che una cosa simile:

```
mov   S0,    %eax
lab:
; istruzioni del ciclo
inc   %eax
cmp   S10,   %eax   ; for
(i=0;i<10;i++)
jl    lab
```

Questo non è molto importante ai fini di quanto andremo a fare nel nostro articolo, ma è sempre interessante sapere che una struttura ad alto livello come il for alla fine non è altro che una serie di poche istruzioni in assembly.

4. INIZIA IL REVERSING

Una volta installato QBob apriamo l'eseguibile che troviamo nella directory in cui l' abbiamo installato con W32Dasm. Quando l'avremo aperto avremo davanti la serie di istruzioni che vengono eseguite dal nostro programma, con a fianco i relativi opcode. Dal momento che abbiamo, come potrete notare, una serie di istruzioni bella lunga, dovremo restringere il campo di azione alle poche istruzioni che

ci interessano.Per questo scopo W32Dasm può fare una cosa molto interessante, ossia mostrarci tutte le stringhe che vengono utilizzate all'interno del programma e i punti in cui vengono utilizzate. Prima di usare questo tool dovremo individuare la stringa che vogliamo utilizzare per trovare il punto del programma in cui intervenire. La stringa ideale è quella che viene mostrata nel messaggio di errore se inseriamo un serial sbagliato. Andando a ricercare dove viene usata quella stringa sapremo dov'è che viene fatto comparire il messaggio d'errore e di conseguenza potremo risalire al punto in cui viene effettuato il controllo del seriale da noi inserito con quello calcolato dal programma. Una volta arrivati a quel punto dovremo modificare l'eseguibile in modo che venga accettato qualunque serial, ma procediamo un passo alla volta. Come abbiamo detto apriamo QBob, dal menù Game scegliamo Register, inseriamo dei dati casuali e otterremo il messaggio d'errore:

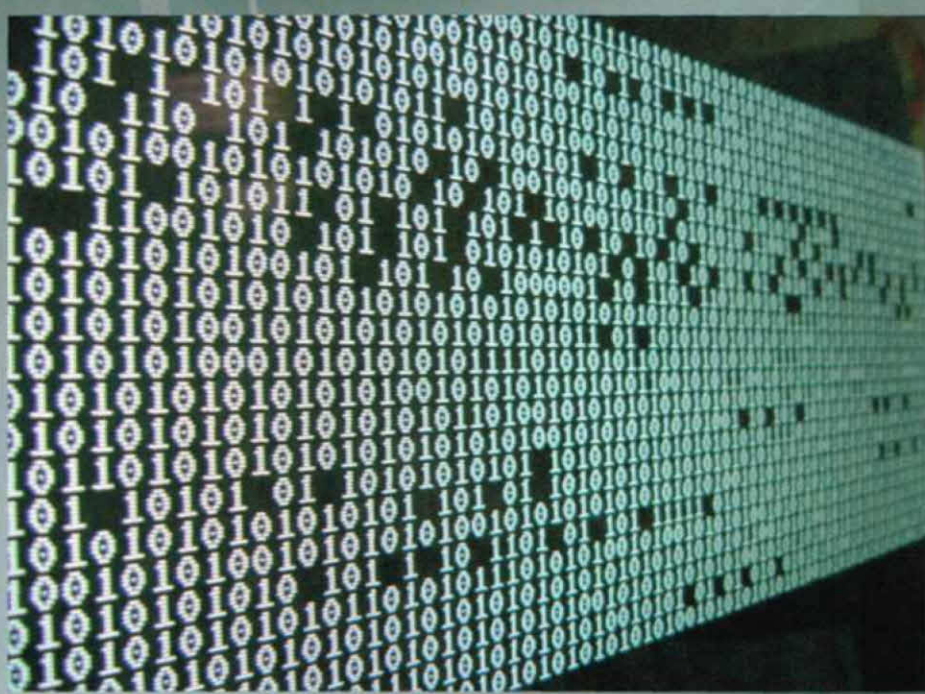
"The serial number is invalid. Please make sure it exactly matches (including dashes) the serial number provided to you by MoonRock Software Inc."

Ora chiudiamo QBob (magari prima facciamoci una partita) e torniamo al disassembler. Andiamo sul menu Refs, quindi String Data References, ossia, come detto in precedenza, il tool che ci consente di risalire alle stringhe usate dal programma. Scorrendo la lista di stringhe troveremo quella che ci interessa, *"The serial number is invalid."* Selezioniamola e verremo direzionati alla parte di codice che utilizza quella stringa:

```
* Referencedbya(U)
nconditionalor(C)
onditionalJumpatAddress:
|:0042EDAA(C)
|
:0042EDD3 8B4660
moveax, dwordptr[esi+60]
:0042EDD6 50                push eax
:0042EDD7 E8F4010000              call
0042EFD0
:0042EDDC 85C0
test eax, eax
:0042EDDE 7527                    jne
0042EE07
:0042EDE0 E802580200call
004545E7
:0042EDE5 8B4010
mov eax, dword ptr [eax+10]
:0042EDE8 6A10
push 00000010
:0042EDEA 50                push eax

*
PossibleStringDataReffromDataObj
>"Theserialnumberisinvalid."-
>"Pleasemakesureitexactlymatches"
>"(includingdashes)theserial"
>"numberprovidedyouby

MoonRock">"SoftwareInc."
|

:0042EDEB 684C214700
                push 0047214C
:0042EDF0 8BCE
                mov ecx, esi
:0042EDF2 E8EFDC0100
                call 0044CAE6
:0042EDF7 5E
        pop esi
:0042EDF8 8B4C240C
                mov ecx, dword
ptr [esp+0C]
:0042EDFC 64890D00000000
                mov dword ptr
```

fs:[00000000], ecx
:0042EE03 83C418
add esp, 00000018
:0042EE06 C3
ret
* Referencedbya(U)
nconditionalor(C)
onditionalJumpatAddress:

I:0042EDDE(C)
I:0042EE07 8B0D082B4700
mov ecx, dword ptr [00472B08]
:0042EE0D894C2408
mov dword ptr [esp+08], ecx
:0042EE11 8B565C
movedx, dword ptr [esi+5C]
:0042EE14 8D442408
leaeax, dword ptr [esp+08]
:0042EE18 52
push edx

*
PossibleStringDataReffromDataObj->"QBobwillberegisteredto'%s'."->"Isitcorrect?"

I:0042EE19 6818214700
push 00472118......

Vediamo chiaramente dove viene utilizzata la stringa. Ora dobbiamo letteralmente salire nel listato per arrivare a capire dov'è che avviene la "diramazione" in cui da una parte viene mostrato l'alert di errore e dall'altra viene registrato correttamente QBob. Nel nostro caso siamo fortunati:

:0042EDDE 7527 jne 0042EE07

Notiamo che avviene una Jump If Not Equal subito prima, e l'indirizzo a cui si viene portati è 0042EE07, che, se andiamo a vedere, si trova subito prima che venga utilizzata la stringa

"QBob will be registered to '%s'. Is it correct?"

il che dovrebbe farci pensare che la "diramazione" che stavamo cercando è proprio quella che abbiamo individuato. Siamo arrivati quindi al punto in cui sappiamo qual è l'istruzione da cui tutto dipende, dobbiamo solo capire come modificarla.

5. RAGIONAMENTO VELOCE SU COSA FARE

Sappiamo ora che le nostre istruzioni sono qualcosa del genere:

```
test eax, eax    ;confronto del serial
jne addr         ;jne a un indirizzo 'addr'

...  ;
...  ;           istruzioni che mostrano il
...  ;           messaggio d'errore
...  ;

addr:
...
...              ;istruzioni che concludono
...              ;la registrazione
...
```

6. CAMBIARE LE ISTRUZIONI

Il nostro scopo, adesso, è quello di cambiare le istruzioni, per farlo abbiamo molte possibilità, eccone un paio:

```
test eax, eax    ;confronto del codice
je/jmp addr      ;con je il programma viene registrato solo se
  ;il serial inserito è sbagliato, con una jmp
  ;il programma viene registrato in ogni caso
...                        ;
...                        ;istruzioni che mostrano il
...
  ;messaggio d'errore
...                        ;
addr:
...
...                        ;istruzioni che concludono
...                        ;la registrazione
...
```

Modificando il jne con un je sarebbe come cambiare da "registra il programma se il serial è corretto" a "registra il programma se il serial inserito non è corretto", usando un jmp invece "registra il programma in ogni caso" in quanto è un salto non condizionato.

L'altra possibilità è quella di inserire il giusto numero di NOP (Not Operation, istruzione che serve a non fare nulla e che ha come opcode 0x90) in modo che non venga eseguito nessun jump e l'esecuzione del programma venga fatta "scivolare" direttamente alla registrazione, facendo qualcosa del tipo:

```
test eax, eax    ;confrontodelcodice
nop              ;
nop              ;
nop              ;In questo modo portiamo il
...
  ;programma direttamente alla fine
nop              ;della registrazione
nop              ;
nop              ;
addr:...
...              ;istruzioni che concludono
...              ;la registrazione
...
```

7. UN BYTE PUÒ FARE LA DIFFERENZA

Delle soluzioni proposte nell'articolo verrà utilizzata la prima, che richiede la sostituzione del jne in un jmp, ma, volendo, si può usare anche il secondo metodo senza problemi o magari trovare altre vie... Basta usare la fantasia Per modificare il jne in una jmp

dobbiamo sostituire l'opcode 0x75 (jne) con 0xEB (jmp) dell'istruzione

:0042EDDE 7527 jne 0042EE07

Mentre 0x27 è il numero di byte di cui saltare (nel caso 39 bytes e, stranamente, 0x0042EDDE (che è l'indirizzo dell'istruzione)+0x02(che sono il numero di byte usati dall'istruzione jne e che non vanno contanti)+0x27 (ossia di quanti byte saltare) ci da proprio 0x0042EE07.

Apriamo il nostro editor esadecimale e ricerchiamo la sequenza di byte:

8B 46 60 50 E8 F4 01 00 00 85 C0

che sono i byte subito prima della jne. Una volta trovati (magari non manualmente) possiamo sostituire il byte seguente (0x75) con 0xEB.

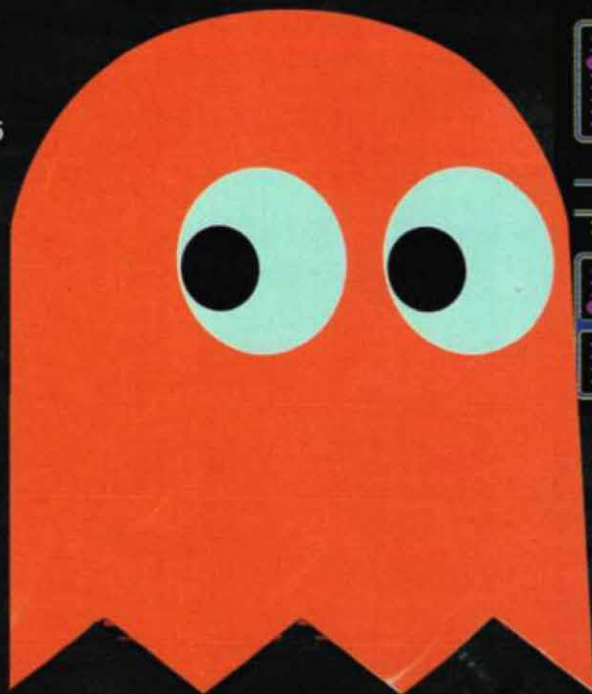
Salviamo e chiudiamo. Ora, per conferma, apriamo W32Dasm e andiamo all'indirizzo di prima:

:0042EDDE EB27 jmp 0042EE07

Tutto come previsto. Avviamo ora l'eseguibile che abbiamo modificato, se non abbiamo fatto nessun errore andando sul menu Game e in seguito su Register possiamo inserire qualunque nome associato a qualunque seriale, avere la nostra copia registrata di QBob, per poter accedere così agli altri livelli etc etc :)

darkjokerhttp://darkjoker.byethost9.com

DISCLAIMER

Gli argomenti e le informazioni fornite nell'articolo sono da considerarsi a scopo puramente informativo. Utilizzare queste informazioni per evitare il pagamento della registrazione è un reato. L'autore e l'editore non si assumono nessuna responsabilità circa l'utilizzo improprio di quanto spiegato.

CYBERENIGMA
HACKER JOURNAL 204

Diciamoci la verità, la crittografia ci ha un po' stufati. Sarebbe il caso di divertirci con qualcosa di nuovo. Durante le cinque parti finora pubblicate del corso di programmazione in C abbiamo dato alcune definizioni relative ai rischi connessi all'utilizzo di determinate metodologie di sviluppo dei nostro software in C.

Per ogni argomento analizzato abbiamo poi corredato la spiegazione con alcuni spezzoni di codice esemplificativo. Obiettivo del Cyber Enigma di questo numero è pertanto lavorare con i sorgenti offerti in tutte le parti del Corso finora pubblicate.

LA SFIDA:

Newbie: Individuare il codice e per ogni blocco vulnerabile individuato spiegare la problematica.
Mid: Correggere il codice vulnerabile facendo uso delle nozioni apprese durante la lettura del Corso.
Esperti: Scrivere almeno un exploit capace di sfruttare eventuali vulnerabilità presenti nel codice presentato in tutte le parti finora pubblicate del Corso.
Guru: Realizzare un exploit per ogni singolo sorgente vulnerabile pubblicato.

SOLUZIONE CYBER ENIGMA HJ 203

La data di nostro interesse in formato GG/MM era ricavabile a partire dal numero della rivista (203), da cui ottenevamo 20 Marzo che, sommando 10 giorni, diveniva 30 Marzo.
Per l'anno andavano mosse alcune considerazioni relative ai suggerimenti indicati, vediamoli uno per uno:
La data dell'evento di nostro interesse (giorno, mese ed anno) è ben più che in evidenza in questo numero.
La data in formato GG/MM era, come visto, riconducibile al numero della rivista. All'anno ci arriviamo tra poco

Dalla luna alle stelle sono sempre presenti e prime in tutto, malgrado non facciano la dieta sono in perfetto peso forma. Meglio non farle arrabbiare che diventano di fuoco e se in cattiva compagnia anche assassine!

Con questo suggerimento ci riferivamo alle molecole di idrogeno (H), primo elemento chimico della tavola degli elementi, il più abbondante e leggero nell'universo ed altamente infiammabile.
Ma noi siamo fortunati, in questo caso sono tenute per mano dalla chiave della vita, fissandosi l'un l'altra come dinanzi uno specchio insieme ad un amico che spesso è talmente generoso da regalare a tutti diamanti.
Questo ci faceva intuire che l'elemento di nostro interesse oltre che idrogeno dovesse contenere anche Carbonio (C) e che la configurazione molecolare dovesse essere simmetrica e separata dall'ossigeno.
Un'esigenza fisiologica di ogni animale (uomo compreso) rappresenta una buona chiave di lettura per questo Cyber Enigma.
Il sonno!
L'oggetto di discussione ormai è da tempo in disuso.
A questo punto una ricerca su Wikipedia con chiave "30 Marzo" con gli elementi appena evidenziati dai suggerimenti ci doveva far riflettere su questo evento:

1842 - L'anestesia attraverso l'uso dell'etere viene usata per la prima volta in una operazione chirurgica dal dottor Crawford Long (cit. Wikipedia).

L'etere dietilico come anestetico è ormai in disuso da un bel po'... ma osserviamo la sua formula: CH3-CH2-O-CH2-CH3.
Notate nulla ? Configurazione simmetrica e molecole di idrogeno in coppia con il Carbonio. Prendiamo quindi in considerazione il numero di queste, l'ossigeno come separatore (eliminando quindi le O dal testo crittato) e scriviamo la chiave come: 3223.
A questo punto consideriamo la stringa crittata e la relativa chiave ed applichiamo quindi una traslazione di -n caratteri:

```
F G T Y H N N R Q Q P D Q G U W H V K C C C V R
3 2 2 3 3 2 2 3 3 2 2 3 3 2 2 3 3 2 2 3 3 2 2 3
C E R V E L L O N O N A N E S T E T I Z Z A T O
```

Complimenti a Luca M. da Roma, l'unico ad aver risolto questo Cyber Enigma alla data di stampa del presente numero (5 Luglio 2010). Cosa aspetti ad iscriverti sul nostro forum ?

Inviate tutto a cyberenigma@hackerjournal.it specificando come oggetto della mail il livello (newbie/mid/esperti) ed il numero della rivista del cyberenigma risolto. I migliori saranno pubblicati in questa pagina e sul sito www.hackerjournal.it!

di Massimiliano Rinaldi
redazione@hackerjournal.itt

FetchMail
la posta da remoto

CLIENT

GESTIRE LA POSTA DA REMOTO IN MODO SICURO È POSSIBILE GRAZIE A FETCHMAIL.

Fetchmail è un MRA (mail retrieval agent) che può servire a scaricare mail da un account remoto sul server locale. Si tratta di un software "made in Linux" piuttosto popolare che può collegarsi a diversi server di posta POP3, IMAP4, ESMTP ETRN e altro, per lo scaricamento dei messaggi mail da molteplici account per poi smistarli sul server locale proprio come fa il postino del circondario che preleva la posta dall'ufficio centrale e poi la smista con, si spera, perizia e precisione.

Fetchmail è configurato nel file $HOME/.fetchmailrc ed è tutto sommato molto semplice da impostare.
Fetchmail va eseguito con l'identità dell'utente destinatario delle mail, mai come root. Se si gestiscono più server di posta, si può aggiungere questa riga.
poll mailserver.yourisp.example protocol pop3 username "foo"

Se il server da cui si sta prendendo la mail supporta IMAP, si può usare imap al posto di pop3. Altre opzioni a disposizione sono password= la password che si vuole usare e ssl.
Per eseguire Fetchmail senza dover inserire una password, basta che questa venga salvata nel file. L'opzione ssl specifica a Fetchmail di connettersi al server tramite SSL/TSL.
Il file .fetchmailrc non deve essere leggibile da terzi. Se accade, Fetchmail solitamente "*se ne risente*". Per assegnare i permessi in modo che nessun altro lo possa leggere, eseguire chmod 0600 $HOME/.fetchmailrc/.

Ma facciamo un passo indietro. L'applicativo può essere scaricato all'indirizzo: http://fetchmail.berlios.de/
Per lanciare Fetchmail basta *digitare:*

```
$ fetchmail
```

Se si vuol eseguire Fetchmail in background, si può usare l'opzione --daemon (oppure -d) insieme a un parametro che gli comunichi ogni quanti secondi *cercare di scaricare le mail dai server:*

```
$ fetchmail --daemon 300
```

Per far partire Fetchmail automaticamente insieme al sistema, aggiungere al file crontab:

@reboot /usr/bin/fetchmail --daemon 300

Fetchmail non può richiedere la password quando viene eseguito in questa modalità. Quindi, affinchè funzioni, è necessario salvare la password in .fetchmailrc. Anche se non si è mai configurato un file crontab *prima, impostarlo è semplice,* basta inserire i tre comandi qui sotto:

```
$ cat > mycron
@reboot /usr/bin/fetch-
mail --daemon 300
<Ctrl+D>
$ crontab mycron
```

Se si gestisce un server IMAP, è possibile offrire accesso via Web alle email installando SquirrelMail (http://squirrelmail.org/, si trova anche nel pacchetto squirrelmail). Iniziare configurando il sistema come un server LAMP e installare e configurare il pacchetto appropriato.

L'applicativo può essere scaricato all'indirizzo: http://fetchmail.berlios.de/

POSTA PROTETTA

Le comunicazioni tra i client di posta e il server spesso contengono informazioni sensibili quindi è meglio abilitare la criptazione SSL/TSL. Per abilitarla in Exime Courier, fare quanto segue:

1) Installare il demone Courier con il supporto SSL/TSL:

```
# apt-get install courier-imap-ssl courier-pop-ssl
```

2) I certificati di CA esterne vengono forniti col pacchetto ca-certificates. Durante la configurazione verrà fatto riferimento al pacchetto, quindi installarlo:

```
# apt-get install ca-certificates
```

Debconf chiede se è preferibile considerare affidabili di default i certificati CA. In molti casi è bene rispondere: "Sì".

3) Se si sta per utilizzare un certificato di una CA sconosciuta (solitamente questo accade solo nel caso che si gestisca una propria CA), inserire il certificato pubblico della CA nel file relativo in /etc/ssl/certs/ e aggiornare il database dei certificati.

```
# update-ca-certificates
```

4) Generare la chiave privata e la richiesta per la firma del certificato. Il posto migliore dove salvare questi file è /etc/ssl/private/. Ecco un esempio:

```
# cd /etc/exim4
# openssl genrsa -out mail.key 1024
# chmod 640 mail.key
# openssl req -new -key mail.key -out mail.csr
# chown root:Debian-exim mail.key
```

5) Far firmare il proprio CSR (Certificate Signing Request) e inserirlo in /etc/mail/private/mail.crt. Per usare un certificato firmato da sé, fare quanto segue:

```
# cd /etc/exim4
# openssl req -new -x509 -nodes -sha1 \
-days 365 -key mail.key -out mail.crt
# chmod 640 mail.crt
# chown root:Debian-exim mail.crt
```

6) Per Courier, concatenare la chiave privata e il certificato in un solo file:

```
# cd /etc/courier
# cat /etc/exim4/mail.key /etc/exim4/mail.crt > mail.pem
# chmod 600 mail.pem
```

7) Abilitare i protocolli SSL/TSL per i demoni IMAP e POP di Courier, editando /etc/courier/imapd-ssl e /etc/courier/pop3d-ssl e sostituendo i valori di TLS_CERTFILE e TLS_TRUSTCERTS con i seguenti:

```
TLS_CERTFILE=/etc/courier/mail.pem
TLS_TRUSTCERTS=/etc/ssl/certs/ca-certificates.pem
```

8)Comunicare a Exim il luogo dove si trovano la chiave privata e il certificato e abilitare anche TLS. Creare il file /etc/exim4/conf.d/main/12_exim4-config_local_tlsoptions, contenente quanto segue:

```
MAIN_TLS_CERTIFICATE = CONFDIR/mail.crt
MAIN_TLS_PRIVATEKEY = CONFDIR/mail.key
MAIN_TLS_ENABLE = 1
```

9) Far ripartire Exim:

```
# invoke-rc.d exim4 restart
```

Ora il server dovrebbe supportare SSL/TSL quando comunica con client SMTP, POP e IMAP.

FIREWALL BUILDER: UNA "FORTEZZA" SU MISURA

MONITORING

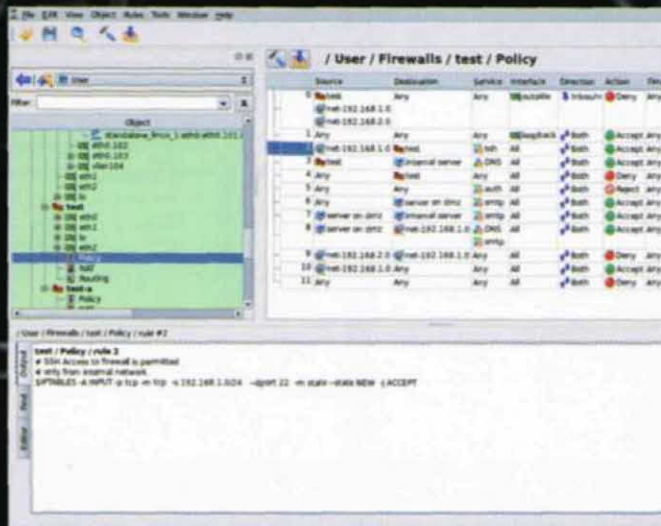
UN OTTIMO STRUMENTO PER PERMETTERE LA MANUTENZIONE DELLE POLICIES DI UN FIREWALL IN MODO SEMPLICE MA EFFICIENTE.

Oggigiorno qualunque rete di computer interconnessa con il mondo esterno (vedi Internet) adotta come primo baluardo a difesa della propria integrità uno o più firewall.
Questi sistemi sono preposti a regolamentare il traffico in entrata e/o in uscita da una rete, assumendo decisioni sulla base della corrispondenza dei pacchetti in transito con decine e decine di regole di filtro.
Ovviamente la gestione di tutte queste regole è, in genere, un' attività abbastanza delicata ed, in quanto tale, fonte di errori che possono riflettersi negativamente sulla sicurezza dell'intero sistema e, quindi, della rete a protezione della quale esso è posto.
A ciò aggiungasi che la complessità delle interfacce a linea di comando e della sintassi utilizzata dagli odierni software di packet filtering costringe ad affrontare una ripida curva di apprendimento e che, una volta acquisita la dimestichezza necessaria, essa può venire del tutto vanificata dall'adozione di un sistema differente.
A fronte di tali difficoltà oggettive esistono però sul mercato strumenti software che permettono di approcciare in modo sicuramente più "friendly" lo svolgimento di tali attività, senza per questo sminuire l'importanza dell'apprendimento.
In particolare uno di tali strumenti va sotto il nome di Firewall Builder ed è disponibile in forma libera per varie piattaforme (Windows, Linux e Mac).
Di esso ci occuperemo in questa sede analizzando la filosofia che ne ha ispirato la realizzazione e soffermandoci sulle sue caratteristiche principali.

CARATTERISTICHE

L'idea di fondo che sta alla base del progetto può essere riassunta in poche parole: permettere la manutenzione delle policies di un firewall in modo semplice ma efficiente.
Per raggiungere tale obiettivo gli autori hanno scelto di adottare la metafora degli oggetti e, grazie all'astrazione in essi insita, sono riusciti a costruire un ambiente tipo RAD molto produttivo che riesce ad agevolare l'amministratore nello svolgimento di una attività, generalmente lunga e tediosa, come quella della creazione delle regole di un firewall.
Grazie a questo differente approccio diventa possibile, in un ambiente interamente basato su GUI, creare un insieme di oggetti che identificano e descrivono i componenti ed i servizi fondamentali di una rete (host, indirizzi, interfacce, sottoreti, protocolli, ecc...) e procedere quindi alla costruzione "visiva" delle policies mediante semplici operazioni di trascinamento o di copia ed incolla.
Una volta implementate le policies vengono memorizzate, insieme con tutti gli altri oggetti, in un file xml e possono essere compilate mediante l'ausilio di un programma esterno che si preoccupa di tradurle nei comandi più adatti producendo uno script di shell oppure un file di configurazione a seconda del tipo di firewall prescelto come piattaforma di deploy.
Altre caratteristiche davvero degne di nota sono inoltre la presenza di oggetti predefiniti che incapsulano le caratteristiche di alcuni tra i più diffusi protocolli e servizi di rete, la possibilità di creare servizi personalizzati ed, infine, il supporto per alcuni tra i più diffusi firewall come iptables, ipfilter, pf e Cisco PIX (per quest'ultimo in realtà è disponibile una licenza commerciale).

INSTALLAZIONE E PRIMA ESECUZIONE

Il funzionamento dello strumento si basa su tre moduli software differenti: una libreria (libfwbuilder), l'interfaccia GUI (fwbuilder) ed un compilatore di policies diverso a seconda del firewall utilizzato come target.
L'installazione (nel caso specifico per Linux ma, come abbiamo visto, Firewall Builder è disponbile anche per

altre piattaforme) è molto semplice: dopo aver scaricato i sorgenti dal sito http://www.fwbuilder.org occorre scompattarli e procedere alla compilazione ed installazione della libreria e, successivamente, della interfaccia GUI mediante le classiche sequenze di configure, make e make install.

Se la compilazione viene portata a termine con successo è possibile lanciare l'interfaccia mediante il comando:

```
fwbuilder &
```

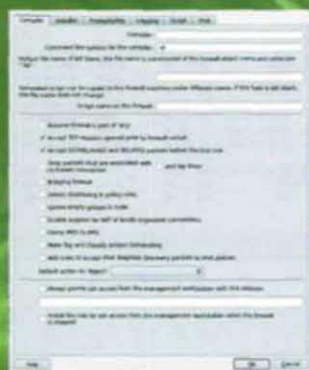
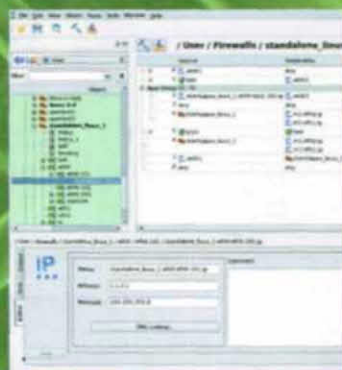
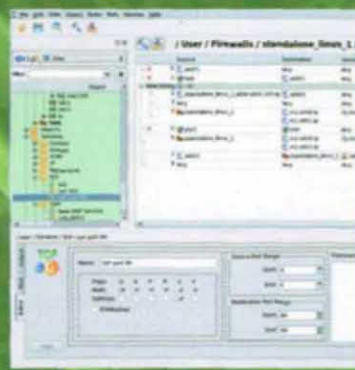
e procedere, quindi, con l'impostazione delle preferenze generali tramite la finestra Options.

I COMPONENTI DI RETE

Come già accennato, una delle funzionalità più apprezzabili di Firewall Builder è costituita dalla possibilità di "inventariare" le componenti fondamentali di una rete in modo da poterle utilizzare durante la successiva fase di creazione delle policies.
La catalogazione degli oggetti può avvenire sia manualmente sia ricorrendo ad una autocomposizione guidata, chiamata Objects Discovery, che permette di utilizzare come metodo di raccolta delle informazioni l'importazione diretta dal file locale hosts, un trasferimento di zona da un server DNS oppure l'utilizzo del protocollo SNMP.
In entrambi i casi avremo comunque a che fare con una serie di entità dal nome molto esplicativo quali host, address, network e firewall.
Un oggetto host è in grado di rappresentare una workstation od un server oppure, più in generale, quasiasi altro nodo della rete munito di un indirizzo proprio.
L'identificazione di ciascun host avviene specificando i suoi parametri fondamentali vale a dire il nome, l'indirizzo IP e l'indirizzo fisico.
In realtà l'indicazione di quest'ultimo è opzionale a meno che non si intenda abilitare a livello di host una interessante caratteristica come quella del filtraggio in base all'indirizzo MAC anzichè IP (ciò richiede però il supporto esplicito da parte del firewall).

Nel caso di sistemi multihomed è inoltre possibile descrivere tutte le interfacce dell'host indicando per ciascuna di esse il nome, l'indirizzo IP, il MAC e la maschera di rete.
Un oggetto address può rappresentare sia un indirizzo IP che un indirizzo MAC e costituisce un attributo fondamentale delle interfacce di rete (ad una stessa interfaccia si possono assegnare più indirizzi IP ma un solo MAC).
Gli oggetti di tipo network consentono di descrivere una rete o sottorete di indirizzi IP mentre gli oggetti address range specificano un range di indirizzi IP contigui.
Tutti gli oggetti censiti possono anche essere raccolti all'interno di entità logiche cd. gruppi e questi ultimi possono essere direttamente richiamati nelle varie regole di filtro.

L'OGGETTO FIREWALL

Questo rappresenta indubbiamente l'entità più complessa ed importante poiché incapsula le caratteristiche dell'intero sistema di packet filtering.
Tale importanza si evince anche dalla presenza di una nutrita serie di pannelli di configurazione attraverso i quali è possibile intervenire direttamente su molteplici aspetti quali:
il tipo di firewall ed il sistema operativo in uso;
le opzioni da utilizzare in fase di compilazione ed installazione delle policies;
le opzioni relative al funzionamento interno del firewall;
le opzioni del kernel del sistema operativo target concernenti alcuni aspetti del funzionamento dello stack TCP/IP;
Analogamente a quanto avviene per gli host un oggetto firewall è tipicamente composto da più interfacce di rete ognuna delle quali viene, a sua volta, identificata in base al nome, all'indirizzo IP e MAC nonchè al tipo.
A tale proposito è bene ricordare che in Firewall Builder le interfacce possono essere di tre categorie

Le opzioni selezionabili.

Editing IP address.

Editing TCP service object.

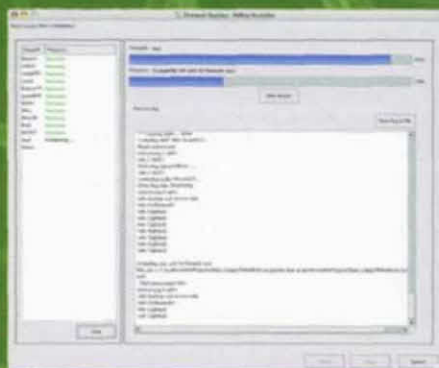
Interactive druid per configurare policies ad hoc.

differenti:

- normali o regolari: sono quelle che hanno un indirizzo IP statico;
- dinamiche: sono quelle che ricevono un indirizzo IP dinamicamente (ad esempio mediante DHCP oppure PPP);
- non numerate: sono quelle che non ricevono mai un indirizzo IP (ad esempio gli endpoint di diversi tipi di tunnel VPN o PPPoE);

Sebbene ogni interfaccia possa avere definite ed associate le proprie policies esiste una differenza fondamentale concernente il loro impiego nella definizione delle varie regole: infatti le interfacce del primo tipo possono essere sempre riferite come sorgente o destinazione dei pacchetti, quelle dinamiche possono esserlo solo se esiste il supporto specifico da parte del firewall (vedi iptables e pf) mentre le interfacce dell'ultimo tipo non possono esserlo mai. Inoltre, diversamente dagli host, le interfacce del firewall possiedono un ulteriore parametro chiamato Security Level che consente di specificare il maggiore o minore livello di sicurezza della stessa interfaccia.

IDENTIFICAZIONE DEI SERVIZI DI RETE

Firewall Builder fornisce una serie di oggetti predefiniti che incapsulano le caratteristiche dei più diffusi protocolli (IP,TCP,UDP e ICMP) e servizi di rete (http,ftp,smtp, ecc...). La completezza e l'abbondanza di tali oggetti non preclude tuttavia la possibilità per l'utente di definire i propri oggetti personalizzati.
Ad esempio, attraverso l'oggetto che rappresenta genericamente il protocollo IP, è possibile costruire altre entità relative a protocolli differenti specificando opportunamente l'informazione relativa al campo di identificazione del protocollo, normalmente presente nell' intestazione dei pacchetti IP. Inoltre lo strumento mette a disposizione anche un servizio definito custom che rende possibile inserire all'interno dello script o del file di configurazione generato per effetto della compilazione qualsiasi opzione valida, secondo la sintassi del firewall utilizzato, che non sia possibile ricomprendere in uno dei servizi predefiniti oppure incorporare in un nuovo servizio.

Configurazione di firewall Cultipli.

LE POLICIES

Ciascun oggetto firewall ha il proprio set di regole che possono essere: globali, relative ad una interfaccia specifica oppure di NAT (Network Address Translation).
Le regole globali e quelle delle interfacce regolano il traffico in entrata e/o in uscita dal firewall sulla base degli indirizzi sorgente e destinazione e di altri parametri.
Le regole di NAT specificano invece le trasformazioni di indirizzi e/o di porte cui devono andare soggetti i pacchetti che transitano nel sistema. Ciascuna di tali regole, a seconda della propria natura, possiede una serie di elementi distintivi che vengono confrontati con gli analoghi parametri dei pacchetti in transito per stabilire una corrispondenza ed intraprendere il tipo di azione prescelta. Le regole globali, così chiamate poichè vengono applicate globalmente, a prescindere dalla interfaccia attraverso la quale transitano i pacchetti, hanno come parametri caratteristici una sorgente, una destinazione, un servizio ed un' azione (generalmente accept, reject o deny). Le regole di interfaccia invece, a differenza delle precedenti, vengono applicate soltanto ai pacchetti che transitano attraverso l'interfaccia per le quali esse sono definite ed hanno gli stessi parametri delle regole globali ma con l'aggiunta di una "direzione" (tale direzione viene ovviamente valutata con riferimento al sistema firewall e non relativamente alla rete protetta dal firewall stesso). Le regole di NAT hanno infine come parametri una sorgente, una destinazione ed un servizio di origine nonchè una sorgente, una destinazione ed un servizio di sostituzione. Intepretazione e modalità di applicazione delle regole I software di packet filtering attualmente esistenti si differenziano, in genere, tra loro per il modo in cui le policies vengono applicate ai pacchetti di rete che li attraversano. In effetti, esistono a tale proposito molte varianti ma è importante sottolineare che in Firewall Builder l'utente lavora sempre con un oggetto firewall astratto e ciò garantisce la più assoluta uniformità di comportamento a prescindere dalla piattaforma prescelta come target. E' lo stesso strumento, attraverso l'uso di appositi algoritmi, a far sì che le regole create nella GUI vengano tradotte nel modo più opportuno e, comunque, tale da garantire il raggiungimento degli obiettivi voluti.
Vale la pena ricordare che Firewall Building oltre a centralizzare ed agevolare le normali attività di amministrazione di un firewall ha anche una valenza didattica. Infatti, una volta che le regole sono state compilate, è sempre possibile analizzare lo script di shell od il file di configurazione prodotto al fine di comprendere come le policies vengano effettivamente tradotte in comandi validi per la piattaforma firewall adottata.

Giovanni Federico - info@giovannifederico.net
Fabio `BlackLight` Manganiello - blacklight@autistici.org

PARTE V

CORSO DI PROGRAMMAZIONE IN C

LINGUAGGI LA QUINTA PARTE DEDICATA A STRINGHE E FILE.

Tutti gli argomenti affrontati di seguito potranno esser chiari solo una volta fatte proprie le nozioni viste nella quarta parte del Corso sui Puntatori. Invitiamo pertanto il lettore a leggere con attenzione la stessa.

STRINGHE

Un tipo di informazione fondamentale nell'informatica consiste nelle sequenze di testo da gestire all'interno di un Calcolatore; tali sequenze vengono denominate stringhe (finora abbiamo infatti gestito solo grandezze di tipo numerico, scalari o vettoriali, non sequenze di testo). Alcuni linguaggi di alto livello, come Java o C# (ma anche lo stesso C++ volendo) gestiscono *le stringhe come tipi di dato astratto a sé stanti*, con i propri metodi e le proprie operazioni. In C le stringhe vengono invece gestite allo stesso modo in cui sono viste dal Calcolatore, ovvero sequenze di caratteri.

Per la gestione delle stringhe quindi non c'è bisogno di tirare in ballo nuovi costrutti o nuovi *tipi di dato, tali entità sono viste allo stesso* modo in cui il Calcolatore gestisce dei semplici array di char. L'inizializzazione di una stringa può essere fatta anche in modo veloce comunque, invece di specificare l'insieme dei caratteri che la compongono separati da virgole, attraverso la seguente scrittura:

```
char str[] = "Corso di programmazione in C";
```

o anche:

```
char *str = "Corso di programmazione in C";
```

Tale scrittura inizializza un array contenente i caratteri 'C', 'o', 'r', 's' eccetera, e, cosa importante, il byte nullo (che il compilatore piazza automaticamente al termine di ogni stringa, volendo rappresentabile esplicitamente dalla sequenza di escape '\0', o semplicemente dal numero intero 0) che rappresenta il termine della stringa.

Il byte nullo è l'unico modo che la macchina ha *per riconoscere la fine di una stringa, e viene* piazzato automaticamente al termine di ogni stringa inizializzata. Un dubbio si dovrebbe essere ora insinuato nella testa dei lettori più svegli: come facciamo quindi a dichiarare una stringa ad esempio binaria, che quindi contenga al suo interno anche il byte nullo senza che questo venga visto come terminatore della sequenza? Ciò è possibile senza grossi problemi, si noti ad esempio la seguente sequenza binaria memorizzata in una stringa:

```
char str[] = "\x00\x01\x02\x00\x01\x02";
```

L'escape '\xdd', dove "d" è una cifra esadecimale (da 0 a 9 o da "a" a "f") identifica un numero esadecimale che può appartenere all'intervallo 00..ff (ovvero da 0 a 255). È evidente in questo caso l'uso del byte nullo (sequenza esadecimale \x00) all'interno della stringa. Tale stringa non può essere gestita da printf, scanf, o uno qualsiasi dei metodi per le stringhe che vedremo fra un attimo, dato che tutti questi metodi gestiscono la stringa identificando il byte nullo come terminatore, ma va gestita manualmente *conoscendo la sua dimensione. Per la stampa* attraverso printf o fprintf di una stringa si usa la stringa di formato "%s", idem, volendo, per la lettura da scanf:

```
char str[] = "Corso di programmazione in C";
printf ("%s", str);
fprintf (stdout, "%s", str);
```

Essendo il primo parametro di printf (e il secondo di fprintf) una stringa, si può anche passare direttamente la stringa come parametro senza specificare la stringa di formato, attraverso una scrittura del genere:

```
char str[] = "Corso di programmazione in C";
printf (str);
fprintf (stdout, str);
```

Tuttavia questa scrittura rende il programma vulnerabile a un attacco tanto pericoloso quanto raffinato, noto come format string overflow *(l'uso del termine overflow, in questo caso, è* fatto per ragioni "storiche", non trattandosi di un autentico overflow). Per capire questo tipo di vulnerabilità si pensi alla scrittura di sopra, e si immagini un contesto in cui "str", invece di essere inizializzata direttamente dal programma, è inserita dall'utente o letta da un file o da *un'interfaccia di rete sotto il controllo dell'utente.* Nulla vieta all'utente di passare str come "%s", o "%d", o "%s, %d, %u" o qualcosa del genere. Se la stringa viene passata in questo modo, funzioni come printf o fprintf semplicemente parsano le sequenze di formato al suo interno e le sostituiscono pescando i rispettivi valori dallo stack del processo. Se i valori da leggere sono passati alla funzione, allora sono questi a essere *stampati, altrimenti è evidente che passando* una stringa di formato costruita appositamente un attaccante può leggere quello che vuole dalla memoria del processo. Si pensi a un caso in cui l'utente inserisce come str qualcosa del genere:

```
char str[] = "%1024x %s";
```

In tal caso decide di voler leggere i 1024 byte *presenti sullo stack di seguito come valori esa*decimali e quello che viene dopo come stringa. Se il processo vulnerabile è un processo di rete e str è passata sulla rete al processo dall'utente, quest'ultimo può leggere arbitrariamente la memoria di un processo residente su un'altra macchina, anche dati che non sarebbe autorizzato *a leggere. La cosa è resa ancora più pericolosa* dall'esistenza della sequenza di formato "%n", che scrive su una variabile intera il numero di byte scritti fino a quel momento dalla (f)printf:

```
int n;
fprintf (stdout, "test\n%n", &n);
fprintf (stdout, "La scrittura precedente ha scritto %d bytes\n", n);
```

Un attaccante può quindi passare stringhe di formato contenenti tale sequenza di formato e,

dosando opportunamente il numero di caratteri che la (f)printf stampa, e sapendo in quale indirizzo di memoria andare a scrivere, può modificare il flusso del programma inserendo salti a locazioni di memoria arbitrarie (magari contenenti uno shellcode infilato in qualche modo). Ci sono strategie molto raffinate per far eseguire con successo al programma vulnerabile a format string overflow del codice arbitrario. Una di queste è quella di scrivere, attraverso la stringa di formato, uno shellcode nel segmento .dtors del processo, dedicato a ospitare i distruttori e con un indirizzo statico (non soggetto alla randomizzazione operata dal kernel), quindi scrivere direttamente all'indirizzo successivo nel code segment un jmp a quella locazione di memoria attraverso l'uso sapiente di %n. Le tecniche in questo caso si fanno molto raffinate e richiedono una trattazione a parte che sarà oggetto di una delle prossime uscite delle rivista.

LINK. UnderAttHack | F. Manganiello - Format String Overflow: http://bit.ly/bGdnCP

Non solo leggere arbitrariamente la memoria di un processo, ma anche scrivere nella sua memoria quindi, ed eventualmente modificare lo stesso flusso di operazioni del processo. È un tipo di vulnerabilità spesso passata in secondo piano rispetto a vulnerabilità apparentemente più diffuse e semplici da sfruttare (lo sfruttamento di un format string overflow come si deve e la scrittura di un exploit sempre funzionante per un programma vulnerabile a esso non è una cosa semplice), ma che è assolutamente da non sottovalutare.

LETTURA

Altro tema delicato è la lettura delle stringhe, ad esempio, da tastiera. L'uso della sequenza "%s" anche per la scanf è ovviamente possibile

```
char str[10];
scanf ("%s", str);
```

Si noti anche che la stringa è passata alla scanf senza il prefisso '&', dato che la scanf vuole un puntatore come secondo parametro, ed essendo una stringa già un array di char, quindi un puntatore a char, non è necessario il prefisso '&'. Ma attenti a questa scrittura. Abbiamo visto poco fa che il compilatore vede il carattere nullo '\0' come terminatore della stringa. Con una scrittura del genere diciamo implicitamente "leggi tutto quello che l'utente passa come stringa finché non preme invio o incontro uno spazio, e salva il risultato in str". Questo è molto pericoloso perché str ha una grandezza finita (10 caratteri in questo caso). Cosa succede se passo da tastiera 100 caratteri? 10 vanno a finire dentro str, gli altri da qualche altra parte. "L'altra parte" è la parte di stack adiacente a str. La cosa può sembrare non così rischiosa, ma di fatto nella parte di memoria dopo str possono esserci altre variabili, che verrebbero sovrascritte dalla lettura incontrollata, e, soprattutto, sullo stack viene salvato anche l'indirizzo di ritorno della funzione, che dice alla funzione a quale indirizzo deve ritornare quando è terminata.
Se riuscissimo a sovrascrivere quell'indirizzo, potremmo modificare senza grossi problemi il flusso del processo, dirottando l'esecuzione all'indirizzo di memoria che vogliamo.
Ancora una volta, se in memoria abbiamo infilato uno shellcode e conosciamo il suo indirizzo, possiamo eseguire sulla macchina il nostro bel codice arbitrario.
Si consideri il caso in cui si scrive un codice come quello sopra e lo si esegue passando al programma 100 'A'.
Quasi sicuramente si avrà un errore di "segmentation fault" una volta eseguito ed il motivo è semplice. Da qualche parte abbiamo sovrascritto l'indirizzo di ritorno della funzione con un indirizzo che non è valido (il codice ASCII della lettera 'A' è 0x41 in esadecimale, se passiamo una valanga di 'A' l'indirizzo di ritorno verrà quasi sicuramente sovrascritto da qualcosa come 0x41414141, indirizzo che nella maggior parte dei casi non è valido e quindi il programma crasha non potendo andare a un indirizzo al di fuori del suo spazio di indirizzamento).
Ma se fossimo in grado di sovrascriverlo con un indirizzo valido, che magari punta al nostro codice arbitrario inserito in memoria o in un file, avremmo il pieno controllo del programma.
Scritture come quella sopra sono quindi da evitare, in quanto sono vulnerabili a quello che è il tipo di attacco probabilmente più famoso nella storia della sicurezza informatica: il buffer overflow. Questo è ancora oggi uno dei tipi di attacchi più diffusi e che miete più vittime, i security bulletin online sono ancora pieni di segnalazioni di questo tipo di vulnerabilità anche nei programmi più diffusi. Sono stati presi molti provvedimenti nel corso degli anni per evitare o minimizzare il rischio di questo tipo di attacchi, fra cui randomizzare l'indirizzo di base dello stack del processo ogni volta che viene eseguito (tecniche di Address Space Layout Randomization adottate dai Windows della serie NT, grosso modo dal 2000 in poi, dal kernel Linux 2.6 e da OpenBSD), in modo da rendere difficile per un attaccante risalire all'indirizzo in cui ha piazzato il suo shellcode, ma questa soluzione è bypassabile attraverso tecniche più raffinate (brute force sullo stack, ret2eip o ret2reg, scrittura dello shellcode in un segmento statico del processo, richiamare direttamente una funzione dalla libc o da una libreria statica invece di infilare uno shellcode arbitrario, e così via).
Le soluzioni moderne si sono fatte via via più raffinate, e includono ad esempio l'azione di rendere il segmento di stack del processo non eseguibile (non c'è in effetti alcun motivo per il quale del codice eseguibile, generalmente presente nel segmento di codice, debba venirsi a trovare sullo stack).
Ma è ancora una battaglia aperta fra i progettisti dei sistemi operativi che cercano di volta in volta di rendere la vita difficile agli attaccanti, e gli attaccanti che ogni volta scovano una strada per bypassare tali misure di sicurezza.
Ancora una volta, la principale causa dei problemi è situata generalmente fra la tastiera e la sedia (in questo caso la sedia del programmatore). Questo tipo di problema sarebbe infatti evitabile se i programmatori effettuassero tutti i controlli necessari sui dati del programma controllabili dall'utente, e si accertassero che il programma non memorizzi dati dell'utente di lunghezza arbitraria dentro zone di memoria di lunghezza finita. Ciò può essere semplice se si ha a che fare con programmi da poche righe di codice, diventa decisamente più impegnativo se si ha a che fare con progetti nell'ordine delle migliaia o milioni di righe di codice. La scrittura di codice sicuro e funzionante in ogni contesto di anomalia dovrebbe essere un qualcosa di innato nello sviluppatore. Il programmatore dovrebbe scrivere nativamente e senza grossi sforzi intellettuali codice che effettui tutti i controlli necessari sui dati manipolati. Ciò tuttavia non accade, spesso anche per colpa del sistema di istruzione che forma i futuri sviluppatori di software che spesso e volentieri fa passare in secondo piano la sicurezza del codice che si scrive in favore della filosofia "l'importante è scrivere codice che funzioni quasi sempre", e non sensibilizza nel giusto modo i programmatori sui reali rischi procurati dal codice insicuro.
È da evitare ancora di più l'uso della funzione gets(), ancora presente in stdio.h ma il cui uso è completamente sconsigliato (tant'è che se si usa gcc si viene esplicitamente avvertiti da un warning al momento della compilazione se il programma contiene una gets). L'uso di gets è molto semplice, basta passare come unico parametro la stringa in cui verrà infilato ciò che viene letto:

```
gets(str);
```

Il suo comportamento è anche magari più desiderabile rispetto a quello della scanf. Quest'ultima infatti interrompe la lettura se all'interno della stringa passata viene incontrato uno spazio, mentre invece la gets legge tutto quello che c'è da leggere finché non viene premuto invio. Tuttavia, non effettua assolutamente nessun controllo sul numero di caratteri da leggere, ed è quindi chiaramente vulnerabile a overflow.
Un buon modo per usare scanf per la lettura delle stringhe è quella di specificare esplicitamente nella stringa di formato quanti byte si vogliono leggere al massimo:

```
char str[10];
```

```
scanf ("%10s", str);
```

Se invece vogliamo leggere una riga intera e non fermarci al primo spazio, possiamo usare la funzione fgets(), che prende i parametri nell'ordine (stringa, lunghezza massima, descrittore file).Vedremo fra poco che tale funzione è utilizzabile anche per leggere da file, e quindi come descrittore del file può essere utilizzato un descrittore aperto che punta a un file, ma nel caso in cui vogliamo leggere da tastiera useremo semplicemente stdin come descrittore:

```
char str[10];
fgets (str, 10, stdin);
str[ strlen(str) - 1 ] = 0;
```

Si noti l'ultima riga. Tale riga è necessaria perché fgets legge anche la pressione di INVIO come un carattere, '\n'. Per rimuovere tale carattere dalla stringa è necessaria una riga che dica "metti a 0 il penultimo carattere della stringa letta". Alcune osservazioni da tenere a mente:

La funzione strlen() ritorna la lunghezza della stringa, e così come le altre funzioni che operano sulle stringhe che vedremo di seguito è utilizzabile a patto che si includa l'header string.h all'inizio del programma.

L'ultimo carattere della stringa è, come già osservato, il byte nullo, quindi il penultimo carattere sarà '\n'.

Settare il penultimo carattere della stringa a 0 vuol dire piazzare il byte nullo, terminatore della stringa, prima, e quindi spostare la fine della stringa.

Entrambi i modi per la lettura delle stringhe esaminati finora consentono di leggere stringhe a patto che sia già nota la dimensione massima. Questo può essere un comportamento non sempre desiderato, vorremmo poter leggere stringhe di lunghezza arbitraria finché non viene incontrato il terminatore o finché non viene premuto invio.

Questo è possibile usando l'allocazione dinamica della memoria e la funzione realloc() vista nell'articolo precedente, scrivendo una funzione del genere:

```
char* get_line() {
        char ch;
/* Stringa, non inizializzata */
        char *line = NULL;
/* Dimensione di partenza della stringa */
        int size = 0;
/* Finché riesco a leggere un carattere dall'input
senza incontrare '\n' o la fine di stdin... */
        while ((ch = getchar()) != '\n' &&
!feof(stdin)) {
/* Incrementa di 1 la dimensione della stringa */
        line = (char*) realloc( line, ++size );
/* Piazza il nuovo carattere alla fine della stringa
*/
        ine[size-1] = ch;
        }
/* Se la stringa contiene almeno un carattere */
        if (line != NULL) {
/* Piazza il terminatore della stringa */
        line[size] = 0;
        }
/* Ritorna la stringa letta */
        return line;
}
...
char *line = get_line();
...
free(line);
```

L'unica funzione usata sopra non ancora nota è feof(). Tale funzione verrà esaminata fra poco quando parleremo dei file, e serve a controllare se è stata raggiunta l'End-Of-File di un certo file. Nel nostro caso, la fine di stdin, che può essere inserita artificiosamente dalla riga di comando dalla sequenza CTRL-D. Un'altra soluzione per leggere stringhe di lunghezza arbitraria è quella di far ricorso alla libreadline, spesso installata di default sulla maggior parte dei sistemi Unix-like, Linux compreso.

La libreadline consente la gestione estremamente potente della lettura da standard input, e offre meccanismi fra cui la gestione dell'history (memorizzazione delle stringhe inserite nel programma) e, settando opportunamente le preferenze, la modifica della riga in stdin da parte dell'utente in un modo che può assomigliare a quello dell'editor Vi o dell'editor Emacs, a seconda dei gusti. La readline si può usare includendo l'header readline/readline.h (ovviamente la libreria deve essere installata sul proprio sistema, su un sistema Debian o "Ubuntu si tratta semplicemente di installare il pacchetto libreadline-dev) e richiamando gcc al momento della compilazione con il parametro -lreadline per specificare che si vuole linkare l'eseguibile usando la libreria esterna. Questo procedimento, valido per la readline, è lo stesso ogni volta che si vogliono usare nel proprio programma librerie esterne. L'uso della readline, una volta fatti questi "preliminari", è estremamente semplice:

```
char *line = readline("Inserisci una stringa: ");
```

L'header string.h, presente in ogni ambiente di sviluppo C, ha molte funzioni per la gestione delle stringhe. Fra queste si annoverano:

Copia, attraverso la funzione strcpy():

```
char s1[] = "Test";
char s2[10];
/* Copio in s2 il contenuto di s1 */
strcpy (s2, s1);
```

Anche l'uso di questa funzione è pericoloso. Prima di richiamarla bisogna, di nuovo, assicurarsi che non si stiano per copiare dentro la stringa di destinazione più caratteri di quelli che può contenere. Altrimenti, si può usare la sua variante "sicura" strncpy(), che prende come terzo parametro il numero massimo di byte da copiare:

```
char s1[] = "Test";
char s2[10];
/* Copio in s2 il contenuto di s1, massimo 10
bytes */
strncpy (s2, s1, 10);
```

Lunghezza: la lunghezza di una stringa si ottiene, come già visto, usando la funzione strlen(stringa) Concatenamento, attraverso la funzione strcat():

```
char str[20] = "Test";
/* Dopo la chiamata della funzione, s2 conterrà
"Test prova" */
strcat (str, " prova");
```

Ancora una volta, questa funzione è a rischio overflow se non si fanno controlli sul numero di caratteri letti.

La sua alternativa sicura è strncat():

```
char str[20] = "Test";
/* Dopo la chiamata della funzione, s2 conterrà
"Test prova" */
strncat (str, " prova", 20);
```

Scrittura attraverso stringhe di formato, attraverso la funzione sprintf() (definita in stdio.h, a differenza delle altre):

```
char str[100];
char nome[] = "Mario";
char cognome[] = "Rossi";
int eta = 30;
sprintf (str, "Mi chiamo %s %s e ho %d anni",
nome, cognome, eta);
```

Anche questa funzione è evidentemente vulnerabile a overflow. L'alternativa sicura è snprintf():

```
char str[100];
char nome[] = "Mario";
char cognome[] = "Rossi";
int eta = 30;
snprintf (str, 100, "Mi chiamo %s %s e ho %d
anni", nome, cognome, eta);
```

Le funzioni viste sopra funzionano a patto che operino su stringhe di testo, o stringhe che non contengono al loro interno il byte nullo. Il funzionamento è infatti basato sul principio che il byte nullo identifichi la fine della stringa. Ovviamente può capitare spesso di gestire stringhe binarie, che in quanto tali possono contenere al loro interno il byte nullo, e in questi casi la maggior parte delle funzioni di string.h diventa inefficace. Si usano in questo caso altre funzioni:

Inizializzazione, attraverso memset():

Tale funzione inizializza tutti gli elementi di una stringa a un valore. Ad esempio, per settare tutti i valori della stringa a 0:

```
char str[100];
memset (str, 0, 100);
```

Tale operazione è anche possibile attraverso la scrittura

```
char str[100] = {0};
```
Copia, attraverso la funzione memcpy():
```
char s1[] = "\x00\x01\x02\x03\x04";
char s2[10] = { 0 };
/* Copio i 5 byte di s1 dentro s2 */
memcpy (s2, s1, 5);
```

FILE

Poter disporre di strumenti capaci di manipolare informazioni sottoforma di file è una caratteristica di fondamentale rilievo in qualsiasi contesto applicativo. Iniziamo col dare alcune prime definizioni formali che saranno poi snocciolate e tradotte in codice C durante il prosieguo.

DEFINIZIONE 25

La particolare importanza della struttura file è da ricercarsi nel fatto che essa non deve necessariamente indicare la propria dimensione all'atto della definizione ed i valori in essa contenuti sono allocati in una delle memorie di massa disponibili nel sistema entro cui è sviluppato l'applicativo. Dal momento che le informazioni contenute all'interno dei file devono essere manipolate e gestite rispettando il modello Architetturale di Von Neumann (vedi "Richiamo Teorico 1" - I Parte), a questi sono fatti corrispondere dei buffer in memoria centrale capaci di consentire il transito di dati da e per la memoria di massa. Nella fase di lettura il buffer è quindi letto dalla memoria di massa ed inserito nel buffer, viceversa, nelle operazioni di scrittura esso viene dapprima inserito nel buffer per poi essere trascritto nella memoria di massa. Le operazioni di accesso ai file possono essere di tipo diretto oppure sequenziale: nel primo caso si accede ad un particolare dato contenuto nel file essendo nota la sua posizione, nel secondo, invece, l'accesso avviene prelevando di volta in volta un valore dopo l'altro.

Si definiscono inoltre determinate operazioni su file, le quali devono essere fin da ora ben chiare al programmatore. Anche qui vedremo poi come queste si traducano in codice C. Quanto finora espresso è sufficiente per capire che avere la possibilità di disporre di un sistema di memorizzazione delle informazioni persistente (ovvero non legato al momento entro cui è avviato il nostro applicativo) rispetto alle operazioni finora analizzate nel corso che facevano uso della memoria volatile del Calcolatore sia un aspetto di notevole rilievo ed utilità per il programmatore. Un file è quindi in buona sostanza "un pacchetto di informazioni" trascritte in modo persistente sul disco.

DEFINIZIONE 26

Per definizione la struttura file deve consentire le seguenti operazioni: Inserimento ed estrazione di elementi; cancellazione, riavvolgimento e azzeramento del file; supporto al predicato logico EOF (End of File) per individuare la terminazione del file.

Queste informazioni, ed è qui snodata la vera utilità, possono essere di qualunque tipo: dal testo alla musica, dalle immagini ai video. Assumono inoltre concreta rilevanza in tutti quei contesti in cui vi sia necessità di elaborare informazioni di archivio in quanto possono costituire uno strumento valido per la realizzazione di database di dati aggregati secondo predeterminati schemi. Come facilmente intuibile leggendo il "Richiamo Teorico 10", per poter adoperare un file è necessario comunicare l'intenzione di farlo al Sistema Operativo. Questo, come altrettanto intuibile, però non basta. Dobbiamo infatti specificare in che modo si intende adoperare ed accedere al file. Predette modalità possono essere: Apertura e posizionamento ad inizio file per operazioni di lettura;
apertura ed eventuale azzeramento del file ove preesistente per operazioni di scrittura;
apertura e posizionamento a fine file per operazioni di scrittura (aggiunta dati);
apertura e posizionamento ad inizio file per lettura e scrittura;
apertura ed eventuale azzeramento del file ove preesistente per orazioni di lettura e scrittura;
Utilizzando l'ormai nota al lettore libreria stdio.h la funzione di nostro è rappresentata dalla fopen(), per la quale offriamo la sua definizione:
```
FILE* fopen(const char* nome_file, const char* modo_accesso);
```
Il primo parametro della funzione è rappresentato da un puntatore a stringa che indica il nome del file da aprire comprensivo del suo percorso sulla macchina , in assenza di ciò sarà letto o creato nello stesso percorso occupato dall'eseguibile. Il secondo parametro indica invece la modalità di apertura del file e può essere, fedelmente a quanto succitato: "r" (lettura), "w" (scrittura, il file è azzerato se preesistente), "a" (aggiunta valori a file preesistente), "r+" (lettura e scrittura), "w+" (lettura e scrittura con azzeramento file ove preesistente).
Questi parametri sono quelli da adoperare nel caso in cui si utilizzino file di testo. Per operare con file binari di qualunque tipo è sufficiente semplicemente aggiungere il carattere "b" al parametro di nostro interesse. In modo duale, al termine delle operazione sul file, utilizzeremo la funzione fclose(), anch'essa facente parte della libreria standard Input/Output del C (stdio.h) e la cui definizione è la seguente:
```
int fclose(FILE* stream);
```
L'unico parametro della stessa è rappresentato da un puntatore alla struttura FILE che analizzeremo a breve. Nella sesta parte del corso saranno poi analizzati nel dettagli i concetti alla base delle strutture dati in C. Analizziamo e commentiamo riga per riga un primo elementare sorgente che scrive su un file di testo di nome "test.txt" la frase "Quinta parte del Corso di Programmazione in C della rivista.":
```
#include <stdio.h>
int main() {
FILE *file;
file = fopen("test.txt", "w");
fprintf(file, "Quinta parte del Corso di Programmazione in C della rivista.\n");
fclose(file);
return 0;
}
```
Alla riga 3 notiamo il richiamo alla struttura FILE puntata al nome, del tutto arbitrario, "file". Alla quarta ci riferiamo alla stessa ed usando la funzione fopen() apriamo in scrittura (w) il file "test.txt".
Alla quinta riga utilizziamo

STRCPY & STRCAT: THE OPENBSD WAY

RICHIAMO TEORICO 10

Come appena visto strncpy() e strncat() rappresentano le due funzioni ANSI C preposte rispettivamente alla copia ed al concatenamento sicuro di stringhe. Esistono tuttavia due diverse implementazioni che non fanno parte dell'ANSI C sviluppate da Todd C. Miller e Theo de Raadt inizialmente per il sistema operativo OpenBSD ma di seguito rese disponibili anche per Linux, Solaris, FreeBSD e Mac OS X.

Esse sono strlcpy() e strlcat(), rispettivamente copiano e concatenano stringhe garantendo la presenza del byte nullo nella stringa di destinazione e ritornando l'esatta dimensione della stringa originata (non semplicemente la sua lunghezza). Di seguito le loro definizioni:

```
size_t strlcpy(char *dst, const char *src, size_t dim);
size_t strlcat(char *dst, const char *src, size_t dim);
```

la funzione fprintf() (stravista in questo corso di programmazione ma adoperata finora a discapito della più elementare printf() per scrivere sullo standard output) per dirigere quanto passato come secondo parametro allo stream "file". Alla riga 6, ultimate le operazione sul file, lo chiudiamo adoperando la funzione fclose(). Infine, come di prassi ormai, compiliamo ed osserviamo l'output:

```
$ gcc scrittura.c -o scrittura
$ ./scrittura
$ cat test.txt
Quinta parte del Corso di Programmazione in C della rivista.
```

Vediamo ora un sorgente che faccia esattamente l'opposto, ovvero leggere e stampare a video il contenuto del file "test.txt":

```
#include <stdio.h>
int main() {
char buffer[100];
FILE *file;
file = fopen("test.txt", "r");
fgets(buffer, 100, file);
printf("%s", buffer);
fclose(file);
return 0;
}
```

Alla riga 9 di questo sorgente facciamo una nuova interessante scoperta. La funzione fgets() ci consente di leggere il contenuto del file trascrivendo n-1 caratteri in un buffer precedentemente allocato.

Osserviamo la sua definizione:

```
char fgets(char* buffer, int n, FILE *stream);
```

Il primo parametro costituisce il punto di arrivo dei caratteri letti dal file, il secondo il numero di caratteri che si intende leggere ed il terzo la struttura FILE di riferimento. Compiliamo ed osserviamo l'output.

```
$ ./lettura
Quinta parte del Corso di Programmazione in C della rivista.
```

Duale della funzione fgets() è la fputs(), la quale si occupa di scrivere una stringa nel file e la cui sua definizione è la seguente:

```
int fputs(const char* str, int n, FILE *stream);
```

Per testarla, scriviamo e compiliamo il seguente sorgente esemplificativo:

```
#include <stdio.h>
int main(){
    char *stringa = "Hacker Journal 204\n";
    FILE *fp;
    fp = fopen("test2.txt", "w");
    fputs(stringa, fp);
    fclose(fp);
    return 0;
}
```

Il funzionamento dello stesso dovrebbe essere a questo punto alquanto chiaro al lettore, osserviamo pertanto direttamente l'output dell'applicativo:

```
$ gcc fputs.c -o fputs
$ ./fputs
$ cat test2.txt
Hacker Journal 204
```

Consideriamo ora su un ulteriore esempio che ci permetterà di scoprire la duale di fprintf() ed ovvero la funzione fscanf() che, a differenza della prima, ci permette di leggere dallo stream piuttosto che scrivere.

La definizione è la seguente:

```
int fscanf(FILE *stream, char *formato, *dest, ...);
```

Scriviamo su un file di testo due numeri interi. Obiettivo del nostro applicativo sarà leggere il file, individuare gli interi, sommarli, restituire a video e salvare sul file precedentemente utilizzato in lettura il risultato come terzo numero.

```
#include <stdio.h>
int main(){
        int a, b, somma;
        FILE *fp;
        fp = fopen("somma.txt", "r+");
        fscanf(fp, "%d %d", &a, &b);
        somma = a + b;
        fprintf(stdout, "%d + %d = %d\n", a,
b, somma);
        fprintf(fp, " %d\n", somma);
        fclose(fp);
}
```

Alla riga 5 invocando la ormai nota fopen() abbiamo aperto il file "somma.txt" in lettura e scrittura. Con la fscanf() alla riga successiva abbiamo poi estratto i primi (ed unici nel nostro caso) due valori interi andandoli a destinare nelle variabili di tipo intero "a" e "b". A questo punto il risultato della loro somma è stato salvato nella variabile "somma" (riga 7) e stampato a video alla ottava riga. Alla riga 9 abbiamo poi scritto il valore contenuto in "somma" nel file "somma.txt" concatenandolo al contenuto già presente. Infine, alla decima riga, abbiamo chiuso il file. Non ci resta che compilare ed osservare l'output:

```
$ gcc fscanf.c -o fscanf
$ cat somma.txt
200 4
$ ./fscanf
200 + 4 = 204
$ cat somma.txt
200 4 204
```

Torniamo ora al primo sorgente visto in questo paragrafo e modifichiamolo come di seguito:

```
#include <stdio.h>
#include <string.h>
int main() {
        FILE *file;
        char *str = "Quinta parte del Corso di
Programmazione in C della rivista.\n";
        int len = (int)strlen(str);
        int i;
        file = fopen("test.txt", "w");
        for (i = 0; i < len; i++){
                fputc(str[i], file);
        }
        fclose(file);
        return 0;
}
```

Alla riga 10 troviamo la funzione fputc() che, collocata in un for che cicla la stringa "str" carattere per carattere, dispone questi ultimi uno per uno nello stream "file" (test.txt). L'utilizzo della stessa diviene quindi particolarmente utile quando si deve lavorare su "singoli pezzi" del file. Offriamo pertanto la sua definizione:

```
int fputc(int carattere, FILE *stream);
```

Osserviamo quindi l'output del nostro applicativo:

```
$ gcc scrittura.c -o scrittura
$ ./scrittura
$ cat test.txt
Quinta parte del Corso di Programmazione in C della rivista.
```

RICHIAMO TEORICO 11

FILE SYSTEM

La gestione delle memorie di massa è affidata al Sistema Operativo. Esistono tuttavia molteplici modi di catalogare ed organizzare le informazioni in esse contenute sia dal punto di vista logico che fisico (tipologia di supporti hardware). In questa sede basti sapere che le memorie di massa sono gestite come aggregati di informazioni tra loro omogenee a cui vien dato il nome di "file". Ad occuparsi di organizzarli e catalogarli è il file system le cui funzioni variano da tipologia a tipologia ma sono universalmente riconducibili a creazione e cancellazione di file, gestione degli accessi ai dati contenuti, organizzazione dello spazio delle memoria di massa e protezione dei file in base a meccanismi di accesso predefiniti.

Ogni file è inoltre caratterizzato nella totalità dei file system da un proprio nome, da un indice informativo relativo all'allocazione dello stesso sul supporto considerato, dalla dimensione, dalla data di creazione e ultima modifica e dalle informazioni di controllo sui privilegi per le operazioni di lettura, scrittura ed esecuzione.

Del tutto duale è il comportamento della funzione fgetc() che si occupa di leggere un carattere alla volta e restituirlo. Di seguito la sua definizione:

int fgetc(FILE *stream);

Vediamola in azione riscrivendo il secondo sorgente visto nel paragrafo adattandolo per utilizzare la fgetc():

```
#include <stdio.h>
int main() {
        FILE *file;
        file = fopen("test.txt", "r");
        while(!feof(file)){
                printf("%c", fgetc(file));
        }
        printf("%s", buffer);
        fclose(file);
        return 0;
}
```

Da segnalare la novità introdotta alla riga 5 dove incontriamo la funzione feof(). Questa si occupa di individuare il predicato logico EOF e quindi la fine del file. Collocandola nel while ed anteponendo l'operatore di negazione "!" ripetiamo il blocco istruzioni del ciclo fino alla fine del file, carattere per carattere. Alla sesta riga, quindi, stampiamo a video ogni carattere attraverso la fgetc(). Concludiamo offrendo uno specchietto delle principali funzioni della libreria stdio.h analizzate in questo paragrafo per lavorare con i file con le relative definizioni ed utilizzi.

I file di testo non sono naturalmente l'unica possibilità di memorizzazione permanente su disco offerta dal C al programmatore. È infatti possibile lavorare, allo stesso modo di quanto finora visto, con file binari eseguendo con quest'ultimi le stesse operazioni di scrittura e lettura. Oggetto di nostro interesse in quest'ottica sono due funzioni della libreria stdio.h: fread() e fwrite(). Prima di procedere con la definizione delle stesse è opportuno però dare un'occhiata al Richiamo Teorico 11.

Offriamo pertanto le rispettive definizioni delle funzioni appena viste:

size_t fread(void *buffer, size_t dim, size_t quant, FILE *stream)

size_t fwrite(void *buffer, size_t dim, size_t quant, FILE *stream)

RICHIAMO TEORICO 12

LETTURA E SCRITTURA DI FILE BINARI

L'accesso ai dati contenuti all'interno di un file in modalità binaria avviene a livello di byte. Da qui le operazioni di scrittura e lettura dipendono da una posizione riferita in memoria (un puntatore), dalla dimensione del blocco dal quale attingere i dati, dalla quantità di informazione da voler leggere/scrivere e dalla dimensione del buffer destinatario.

Con la prima possiamo leggere il contenuto del file indicato dal puntatore allo stream (FILE *stream), trascrivendolo all'interno di un array (void *buffer). La dimensione degli elementi da copiare nel buffer (espressa in byte) deve ovviamente rispettare la dimensione degli indici dell'array ricevente (size_t dim). La quantità di elementi di "dim" dimensione da trascrivere nel buffer è infine indicata dal programmatore (size_t quant). In parole povere, la funzione fread() legge complessivamente quant * dim byte da file copiandoli nel buffer indicato come primo argomento. Con la seconda, viceversa, scriviamo il contenuto dell'array (void *buffer) nel file indicato (FILE *stream). Come per la fread(), la dimensione degli elementi da copiare nel file è espressa in byte (size_t dim) e la quantità è indicata dal programmatore all'atto della chiamata della funzione (size_t quant). Saranno pertanto prelevati dal buffer (primo argomento) quant * dim byte e scritti nel file indicato come quarto parametro.
È importante precisare che entrambe le funzioni trattano, come prevedibile, sequenze di byte. Esse, in quanto tali, non sono interpretabili e possono rappresentare qualunque informazione (immagini, video, audio, testo).
Nel sorgente che segue abbiamo testato il funzionamento di entrambe le funzioni richiedendo da tastiera 10 interi da collocare in un primo array. Abbiamo quindi copiato il contenuto dello stesso all'interno di un file, utilizzando successivamente quest'ultimo in lettura per ricollocare gli interi all'interno di un nuovo array.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int array_r[10] = {0};
    int array_w[10] = {0};
    int i;
    FILE *fp_in;
    FILE *fp_out;
    for(i=0; i<10; i++){
        printf("Inserire numero #%d: ", i+1);
        scanf("%d", &array_r[i]);
    }
    printf("Contenuto ARRAY_R: ");
    for(i=0; i<10; i++){
        if(i == 9)
            printf("%d\n", array_r[i]);
        else
            printf("%d, ", array_r[i]);
    }
// Trascrivo il contenuto dell'array di interi su file:
    fp_in = fopen("store", "wb");
    fwrite(array_r, sizeof(int), 10, fp_in);
    fclose(fp_in);
// Ora leggo da file e trascrivo su array:
    fp_out = fopen("store", "rb");
    fread(array_w, sizeof(int), 10, fp_out);
    fclose(fp_out);
// Stampo a video il contenuto del nuovo array:
    printf("Contenuto ARRAY_W: ");
    for(i=0; i<10; i++){
        if(i == 9)
            printf("%d\n", array_w[i]);
        else
            printf("%d, ", array_w[i]);
    }
    return 0;
}
```

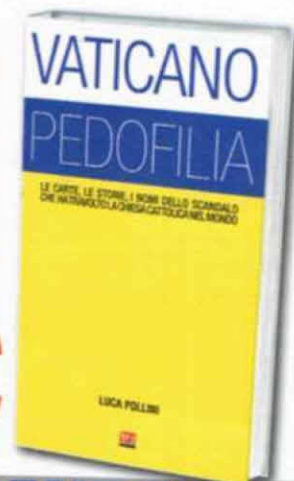
Osserviamo l'output:

```
$ gcc bin.c -o bin
$ ./bin
Inserire numero #1: 1
Inserire numero #2: 2
Inserire numero #3: 3
...
Inserire numero #10: 10
Contenuto ARRAY_R: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Contenuto ARRAY_W: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

Si badi alla differenza nel file generato dall'applicativo che, a differenza degli esempi visti prima con file di testo, è binario. Attestiamo la cosa utilizzando il comando "file":

```
$ file store
store: data
$ file test.txt
test.txt: ASCII text
```

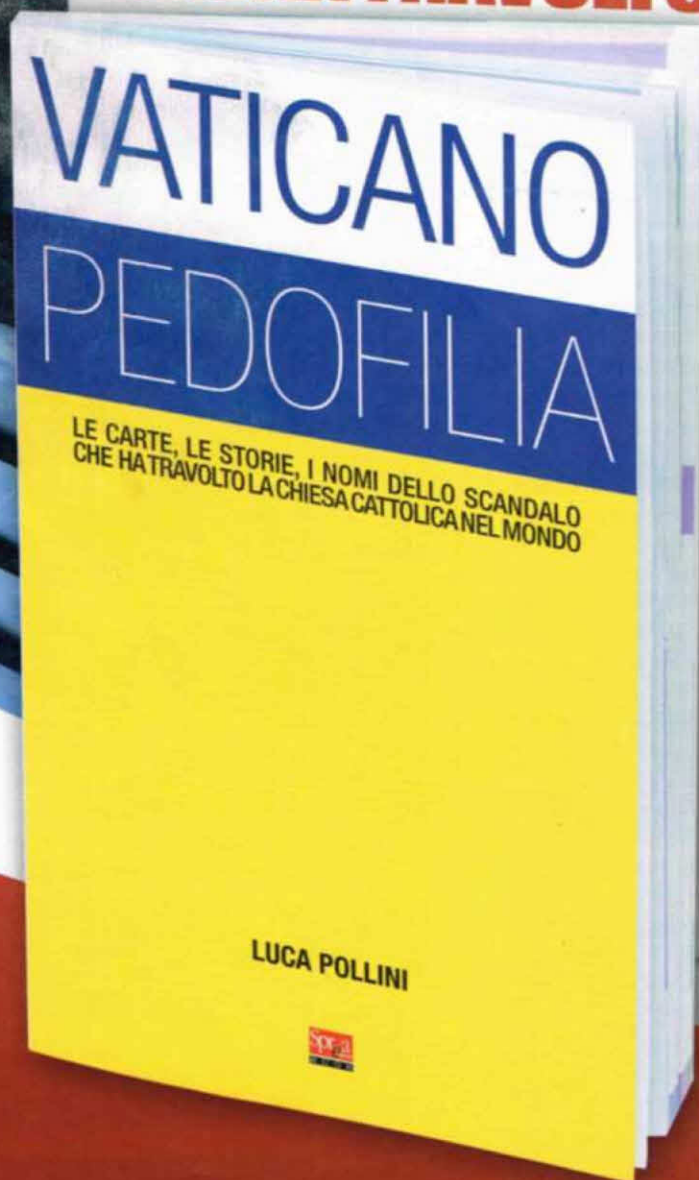
FUNZIONE	DEFINIZIONE	UTILIZZO
fopen()	FILE* fopen (const char * filename, const char * mode);	Apertura file.
fclose()	int fclose (FILE * stream);	Chiusura file.
fprintf()	int fprintf (FILE * stream, const char * format, ...);	Scrittura formattata su stream.
fscanf()	int fscanf (FILE * stream, const char * format, ...);	Lettura formattata da stream.
fgets()	char* fgets (char * str, int num, FILE * stream);	Lettura di num caratteri da file.
fputs()	int fputs (const char * str, FILE * stream);	Scrittura di str su file.
fgetc()	int fgetc (FILE * stream);	Lettura carattere da file.
fputc()	int fputc (int character, FILE * stream);	Scrittura carattere su file.
feof()	int feof (FILE * stream);	Indicatore End Of File.

Lo *SCANDALO* che ha scosso il *MONDO!*

IL LIBRO INCHIESTA SULLO SCANDALO CHE HA TRAVOLTO IL VATICANO

VATICANO

PEDOFILIA

LE CARTE, LE STORIE, I NOMI DELLO SCANDALO CHE HA TRAVOLTO LA CHIESA CATTOLICA NEL MONDO

LUCA POLLINI

L'espresso
«SCACCO AL PAPA»

The New York Times
«UN PAPA PUÒ DIMETTERSI?»

DER SPIEGEL
«RATZINGER: (IN)FALLIBILE»

The Washington Post
«COME E PIÙ DEL WATERGATE»

«BISOGNA CAMBIARE PAPA?»

Süddeutsche Zeitung
«LA CHIESA HA NASCOSTO»

IN EDICOLA